UNIVERSIDADE DE LISBOA

FACULDADE DE CIÊNCIAS

DEPARTAMENTO DE FÍSICA

Interface Ethernet para um Testador de Sistemas Electrónicos do Tilecal

José Domingos Resende Gomes Lopes Alves

Mestrado em Engenharia Física

2012

UNIVERSIDADE DE LISBOA

FACULDADE DE CIÊNCIAS

DEPARTAMENTO DE FÍSICA

Interface Ethernet para um Testador de Sistemas Electrónicos do Tilecal

José Domingos Resende Gomes Lopes Alves

Dissertação orientada pelos Profs. Doutores Guiomar Evans e José Soares Augusto

Mestrado em Engenharia Física

2012

Abstract

The present work was developed in the scope of an update of a tester of the electronics systems of the hadronic calorimeter *TileCal* of the ATLAS experiment at CERN. This tester, MobiDICK 4, communicates with a user's computer via an *Ethernet* interface implemented in FPGA. Tests of *Ethernet* interfaces were required to determine the one more suitable for the tester. Another functionality of this tester is to check the integrity of data sent by the front-end electronics system of the *TileCal*, so an algorithm to do this task was implemented in the scope of this work.

The *Ethernet* interfaces' tests were performed by implementing a communication system in full-duplex mode in our electronics laboratory, where the *Ethernet* communications between a *Xilinx* ML605 board equipped with a Virtex-6 FPGA, and a computer, using the TCP/IP protocol was tested. Two *Ethernet* interfaces available in the *Xilinx*'s tools were implemented and tested: the *Tri-mode Ethernet Media Access Control* (TMAC) interface and the *Ethernet Lite Media Access Control* (ELM) interface, incorporated in an embedded system controlled by the embedded soft-core microprocessor *MicroBlaze*. The implementation and tests were performed using the *Integrated Software Environment* (ISE) from *Xilinx,* and the resources available in the *Embedded Development Kit* (EDK). EDK allows a simple way to implement a complete and functional embedded system with a microprocessor for deploying in a *Xilinx*'s FPGA.

The data integrity check algorithm was implemented and tested in the control board of the tester, the ML507 board equipped with a Virtex-5 FPGA. This algorithm will be able to check the data integrity in data sent by *TileCal* during tests. After the implementation, validation tests were performed in a real situation of data acquisition from the front-end electronics system of the *TileCal*.

Keywords: TileCal, MobiDICK, Ethernet, Cyclic Redundancy Check.

Resumo

Este trabalho foi realizado no âmbito da actualização de um testador de sistemas electrónicos do calorímetro hadrônico *Tilecal* da experiência ATLAS/CERN. Este testador, o MobiDICK 4, é implementado de forma a comunicar com um computador através de uma interface *Ethernet* implementada numa FPGA, tendo sido necessário efectuar testes às interfaces *Ethernet* disponíveis para este tipo de implementação. Uma das funcionalidades do referido testador é a verificação da integridade de dados enviados pelo sistema electrónico de frontaria instalado no *TileCal*. Foi também implementado, no âmbito deste trabalho, um algoritmo de verificação de integridade de dados chamado de *Cyclic Redundancy Check* (CRC).

O teste das interfaces foi realizado através da implementação de um sistema de comunicação no laboratório de electrónica da FCUL (Faculdade de Ciências da Universidade de Lisboa), operando em modo *full-duplex*, no qual se testou a comunicação *Ethernet* entre uma placa ML605 equipada com uma FPGA Virtex-6 da *Xilinx*, e um computador, com recurso ao protocolo TCP/IP. Foram implementadas e testadas duas interfaces *Ethernet* disponibilizadas pela *Xilinx*: as interfaces *Tri-mode Ethernet Media Access Control* (TMAC) e *Ethernet Lite Media Access Control* (ELM), num sistema embebido controlado pelo microprocessador *soft-core* embebido *MicroBlaze*. A implementação e os testes dessas interfaces *Ethernet* foram efectuados no ambiente *Integrated Software Environment* (ISE) da *Xilinx* com recurso às ferramentas existentes na plataforma *Embedded Development Kit* (EDK). A ferramenta EDK permite a implementação rápida de um sistema embebido completo e funcional, incluindo um microprocessador embebido, para ser configurado numa FPGA da *Xilinx*.

O algoritmo de verificação de dados recebidos pelo testador (enviados pela electrónica de frontaria) foi implementado no CERN, numa placa ML507 equipada com uma FPGA Virtex-5 da *Xilinx*. Esta placa é a placa de controlo do referido testador MobiDICK 4. Após a implementação foram realizadas testes de validação, numa situação real de aquisição de dados da electrónica de frontaria.

Palavras-chave: TileCal, MobiDICK, Ethernet, Cyclic Redundancy Check.

Agradecimentos

À minha família, em especial ao meu pai José Lopes Alves, pelo apoio e motivação que me proporcionou durante todo o meu percurso académico.

À Professora Guiomar Evans pela excelente orientação que me proporcionou durante a realização deste trabalho e também de outros trabalhos orientados por ela, pelo grande apoio e conhecimentos por ela transmitidos durante o meu percurso na FCUL. Agradeço também ao Professor José António Soares Augusto por aceitar co-orientar a realização deste trabalho.

Um agradecimento é dedicado aos meus amigos que apoiaram-me em certos momentos importantes no meu percurso: Justimiano Alves, Edson Ribeiro e Saturnino Borges.

Agradeço também ao Filipe Martins e ao Luís Seabra, pelo apoio que me ofereceram durante a minha estadia no CERN.

Aos elementos da equipa da Universidade de Valência que trabalham no CERN e que prestaram-me todo o apoio durante a realização de parte deste trabalho no CERN, nomeadamente o Alberto Valero, Carlos Solans e Fernando Carriò Argos.

Ao LIP pela atribuição da bolsa de investigação científica, oferecendo-me a oportunidade de integração no grupo de trabalho da colaboração portuguesa na experiência ATLAS/CERN, para a realização deste trabalho.

Conteúdo

Lista de Figuras

Lista de Tabelas

Siglas

ADC	*Analog-to-Digital Converter*
AMBA	*Advanced Microcontroller Bus Architecture*
API	*Application Programmable Interface*
ARP	*Address Resolution Protocol*
ASCII	*American Standard Code for Information Interchange*
ASIC	*Application Specific Integrated Circuit*
ATLAS	*A Toroidal Lhc AparatuS*
AXI	*Advanced eXtensible Interface*
BCID	*Bunch Crossing Identifier*
BSB	*Base System Builder*
CERN	*European Centre Organization for Nuclear Research*
CRC	*Cyclic Redundancy Check*
CSMA/CD	*Carrier Sense Multiple Access/Collision Detection*
DAC	*Digital-to-Analog Converter*
DDR	*Double Data Rate*
DHCP	*Dynamic Host Configuration Protocol*
DIP	*Dual In-line Package*
DMU	*Data Management Unit*
EBCDIC	*Extended Binary Coded Decimal Interchange Code*
EDK	*Embedded Development Kit*
ELM	*Ethernet Lite Media Access Control*
FCS	*Frame Check Sequence*
FPGA	*Field Programmable Gate Arrays*
FSL	*Fast Simplex Link*
FTP	*File Transfer Protocol*
HDL	*Hardware Description Language*
HTTP	*Hypertext Transfer Protocol*
HV	*High Voltage*
ICMP	*Internet Control Message Protocol*
IFG	*InterFrame Gap*
IP	*Internet Protocol*
IP	*Intellectual Property*
ISE	*Integrated Software Environment*
ISO	*International Standards Organization*

LED *Light Emission Diodes*

LHC *Large Hadron Collider*

LIP *Laboratório de Instrumentação e Física Experimental de Partículas*

LLC *Logical Link Control*

LMB *Local Memory Bus*

LPC-CF *Laboratoire de Physique Corpusculaire – Clermont Ferrand*

LPDDR *Low Power Double Data Rate Memory*

LVPS *Low Voltage Power Supply*

LwIP *Lightweight Internet Protocol*

MAC *Media Access Controller*

MDI *Medium Dependent Interface*

MDM *Microprocessor Debug Module*

MHS *Microprocessor Hardware Specification*

MII *Media Independent Interface*

MIIM *Media Independent Interface Management*

MobiDICK *Mobile Drawer Integrity Checking System*

MPMC *Multi-Port Memory Controller*

MSS *Microprocessor Software Specification*

NIC *Network Interface Card*

NLANR *National Laboratory for Applied Network Research*

ODIN *Optical Dual G-Link S-Link*

OSI *Open Systems Interconnection*

PCS *Physical Coding Sublayer*

PING *Packet INternet Groper*

PMA *Physical Coding Attachments*

PMD *Physical Medium Dependent*

PMT *Photomultiplier tubes*

PLB *Processor Local Bus*

POP *Post Office Protocol*

RISC *Reduced Instruction Set Computer*

RMS *Root Mean Square*

ROD *Read Out Drivers*

SDF *Start of Frame Delimiter*

SDK *Software Development Kit*

SDRAM *Synchronous Dynamic Random Access Memory*

SFP *Small Form Pluggable*

SMTP	*Simple Mail Transfer Protocol*
SSP	*Simple S-Link to PMC*
TCP	*Transmission Control Protocol*
TileDMU	*Tile Data Management Unit*
TMAC	*Tri-Mode Ethernet Media Access Control*
TTC	*Trigger Timing Control*
UART	*Universal Asynchronous Receiver Transmitter*
UDP	*User Datagram Protocol*
USB	*Universal Serial Bus*
VHDL	*Very High Speed Integrated Circuits Hardware Description language*
VME	*Versa Module Europa*
WLSF	*WaveLength Shifting Fiber*
XMD	*Xilinx Microprocessor Debugger*
XML	*e**X**tensible **M**arkup **L**anguage*
XPS	*Xilinx Platform Studio*

Capítulo 1 Introdução

O CERN *(European Centre Organization for Nuclear Research)*, fundado em 1954, é o maior centro de investigação em Física de Partículas do mundo. Situa-se na fronteira entre a França e Suíça perto da cidade de Genebra, onde cientistas de vários países estudam a constituição da matéria e as forças que asseguram a sua existência. O CERN dispõe do maior acelerador de partículas da actualidade, o LHC *(Large Hadron Collider)* e de um conjunto de detectores que permitem que tais estudos sejam possíveis.

O LHC representa uma estrutura circular subterrânea com 27 km de perímetro. O principal objectivo do LHC é permitir colisões entre protões que viajam a uma velocidade muito próxima da velocidade da luz, numa tentativa de recriar as condições iniciais do surgimento do Universo ocorridas uma pequena fração de segundo após o *Big-bang*. O LHC permite a colisão entre dois feixes de protões ou iões pesados viajando em sentidos opostos. Os feixes de partículas movem-se no interior do anel do acelerador em tubos nos quais é estabelecido vácuo contínuo. Os protões são acelerados e mantidos no centro dos tubos através de um campo magnético criado por magnetos supercondutores, arrefecidos por um sistema criogénico adequado. As colisões ocorrem em pontos-chave, precisamente onde se situam os detectores de quatro experiências: ATLAS, ALICE, CMS e LHCb.

O calorímetro hadrónico *TileCal* é um dos subdetectores do detector ATLAS do CERN. É utilizado para a medir da energia de jatos de partículas resultantes de colisões protão-protão (que ocorrem durante as experiências realizadas utilizando o LHC), medir a energia transversa em falta e para identificar muões de baixo momento transverso *(low-pt muons)*. Para que isto seja possível, o *TileCal* apresenta um sistema electrónico de aquisição (electrónica de frontaria) rápido e preciso que realiza a aquisição do sinal produzido pelo calorímetro, e um sistema de leitura também rápido e preciso.

Este trabalho foi realizado no âmbito da colaboração Portuguesa na experiência ATLAS, directamente relacionado com o teste da electrónica do seu calorímetro hadrónico, o *TileCal*. Para um melhor enquadramento do trabalho efectuado nesta dissertação, é apresentada seguidamente uma breve descrição do detector ATLAS.

Tal como foi referido, o ATLAS *(A Toroidal Lhc AparatuS)* é um dos detectores instalados no LHC. Apresenta um comprimento de 44 m e uma altura de 25 m, e uma massa de 7000 toneladas [1]. A figura 1.2 ilustra a estrutura deste detector. O ATLAS

constituído por vários subsistemas importantes, cada um deles com características específicas de acordo com a sua funcionalidade [2]:

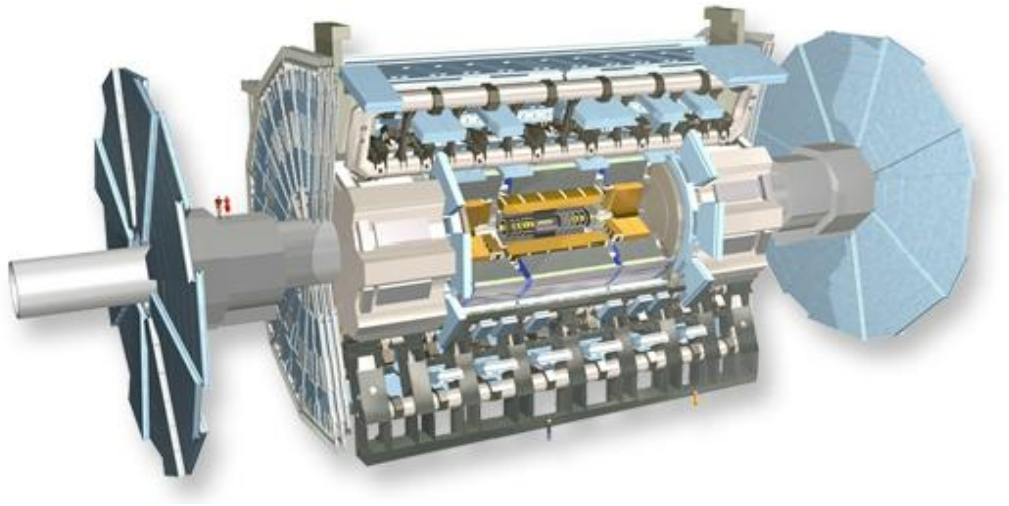


Figura 1.1: *O detector ATLAS.*

O detector ATLAS é constituído pelos seguintes subsistemas principais [1]:

1. **Detector Interno *(Inner Detector)*:** o detector interno mede o momento de partículas carregadas. É implementado para reconstruir trajectórias e vértices (pontos de decaimento) de decaimentos de qualquer evento do LHC, com elevada eficiência.

2. **Calorímetros:** há dois tipos de calorímetros no detector ATLAS, um calorímetro hadrónico *(TileCal)* e um calorímetro electromagnético. Servem para realizar medições precisas da energia e posição de electrões e fotões resultantes dos eventos do LHC.

3. **Espectrómetro de Muões:** utilizado para identificar muões e medir os seus momentos.

4. **Sistema Magnético:** permite encurvar a trajectória de partículas carregadas para se poder medir os seus momentos. A trajectória de uma partícula num campo magnético encurva-se, o raio da curvatura e a direcção permitem determinar o momento e carga da respectiva partícula.

Um sistema electrónico, designado de *Mobile Drawer Integrity Checking System* (MobiDICK), é utilizado para testar as funcionalidades da electrónica de frontaria do calorímetro hadrónico *TileCal*. Este trabalho de dissertação foi desenvolvido no âmbito da actualização deste testador. A actualização deste sistema está a ser realizada por vários

grupos de investigação, de diferentes instituições, incluindo o *Laboratório de Instrumentação e Física Experimental de Partículas* (LIP), que proporcionou as condições para a realização deste trabalho.

No âmbito deste trabalho, a actualização do testador MobiDICK esteve associada à implementação e teste de interfaces *Ethernet* numa FPGA *(Field Programmable Gate Arrays)*.

O novo testador actualmente em desenvolvimento, o MobiDICK 4, tem como base uma versão já existente, o MobiDiCK 3. O MobiDICK 4 é implementado num sistema embebido realizado numa FPGA e comunica com um computador (utilizador) por tecnologia *Ethernet*, com recurso ao protocolo TCP/IP[1]. A escolha da versão da interface *Ethernet* mais apropriada para o MobiDICK 4 motivou as implementações das interfaces *Ethernet* disponíveis no sistema de desenvolvimento da *Xilinx*, que foram configuradas na FPGA. Esses testes visaram o desempenho, funcionalidade, fiabilidade e acessibilidade às interfaces numa comunicação com um computador de teste.

Devido à grande diversidade de sistemas electrónicos que o MobiDICK 4 irá testar, a versatilidade e fácil actualização do testador devem ser asseguradas. Estes requisitos levaram a que o testador devesse ter por base sistemas reconfiguráveis e comunicar com o utilizador através de uma interface *Ethernet*.

Para a implementação e testes das interfaces *Ethernet* foi utilizada a plataforma *Embedded Development Kit* (EDK) da *Xilinx* e as suas ferramentas *Xilinx Platform Studio* (XPS) e *Software Development Kit* (SDK).

Para além desta tarefa, foi implementado e testado um módulo em VHDL[2] que realiza um algoritmo para a verificação da integridade de dados recebidos pelo testador conhecido por *Cyclic Redundancy Check* (CRC). Esta implementação foi efectuada também numa FPGA.

1.1 Motivação

O trabalho realizado nesta dissertação revelou-se bastante motivador, pois permitiu a colaboração numa das importantes experiências do CERN (neste caso a experiência ATLAS), actualmente o maior centro de investigação em física de partículas do

[1] TCP/IP – Sigla de *Transmission Control Protocol/Internet Protocol*
[2] VHDL – Sigla de *Very High Speed Integrated Circuits Hardware Description Language*

mundo. Esta motivação surge, não só pela dimensão e prestígio em termos de investigação e contribuição à Ciência do CERN, mas pelos conhecimentos que se adquire quando se colabora e se trabalha num centro de investigação desta qualidade.

Em relação à implementação e aos testes de interfaces *Ethernet* numa FPGA, conclui-se ser necessário estudar a tecnologia de comunicação *Ethernet* e os protocolos associados. Esta tarefa constituiu uma novidade, pois a formação académica do autor não incluía uma componente que envolvesse telecomunicações, implementação de protocolos (TCP/IP) e aplicações de comunicação (Cliente/Servidor). Foi também necessário aprofundar os conhecimentos da linguagem de descrição de *hardware*, VHDL, e aprender a trabalhar com a ferramenta de desenvolvimento da *Xilinx*, EDK, para o desenvolvimento e implementação de sistemas embebidos (microprocessadores embebidos) em FPGAs.

Foi também necessário estudar e compreender o funcionamento de toda a electrónica de frontaria instalada no calorímetro hadrónico *TileCal*, assim como a arquitectura do testador MobiDICK. O trabalho culminou com o estudo do processo de verificação de integridade de dados implementado no *TileCal* e com a implementação de um módulo de verificação de integridade de dados recebidos pelo testador. Esta implementação foi realizada e testada num dos laboratórios do CERN, em conjunto com outros grupos de investigação que também colaboram na implementação do testador.

Foram vários os conhecimentos novos que o autor necessitou de adquirir, o que fez com que este trabalho fosse bastante motivante; isso aliado ao facto de ser uma oportunidade de colaborar e de conhecer uma das importantes experiências do CERN.

1.2 Plataforma de Desenvolvimento

Nesta secção é introduzida uma descrição da plataforma de desenvolvimento utlizada para a implementação e teste do sistema proposto, a plataforma EDK que faz parte do conjunto de ferramentas disponibilizadas pelo ambiente de desenvolvimento *Integrated Software Environment* (ISE) da *Xilinx*.

1.2.1 Plataforma *Embedded Development Kit*

A plataforma EDK é uma ferramenta que permite o desenvolvimento de sistemas embebidos completos, incluindo microprocessadores, visando a sua implementação numa FPGA da *Xilinx* [3]. A plataforma EDK integra duas ferramentas principais: A *Xilinx*

Platform Studio (XPS) que permite o desenvolvimento da componente de *hardware* e a ferramenta *Software Development Kit* (SDK) focada no desenvolvimento da componente de *software* de um sistema embebido.

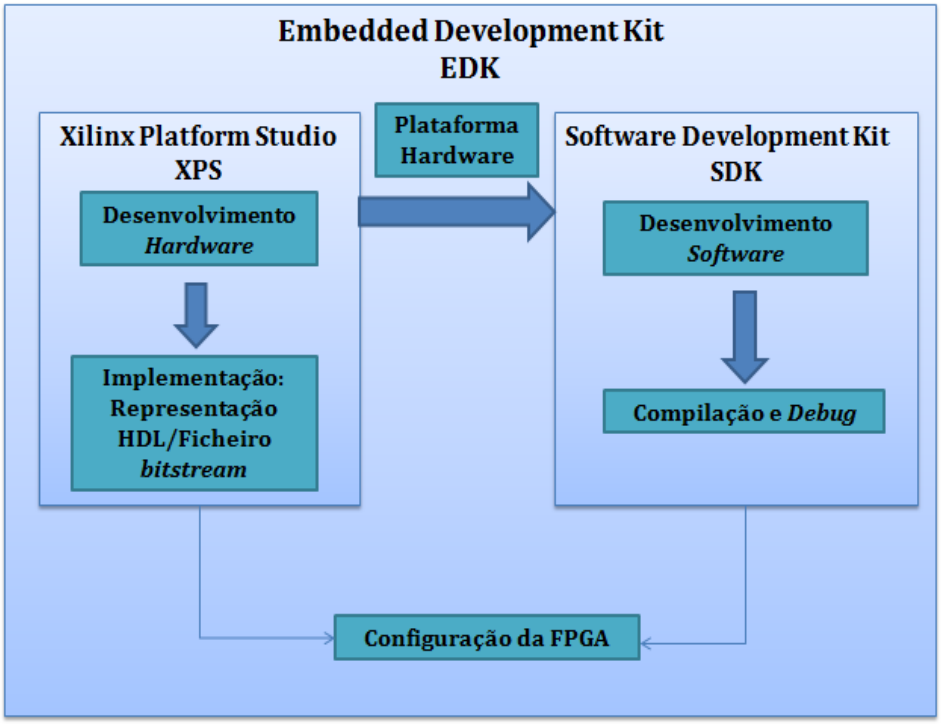


Figura 1.2: *Fluxo de projecto de um sistema embebido através da plataforma EDK.*

A figura 1.2 representa o fluxo de projecto típico quando se utiliza a plataforma EDK, ilustrando os passos fundamentais do desenvolvimento de um sistema embebido com esta ferramenta. A primeira fase é realizada na ferramenta XPS, correspondendo ao desenvolvimento da componente de *hardware*; a seguir passa-se à implementação, que consiste na representação do sistema através de uma linguagem de descrição de *hardware* (VHDL ou *Verilog*) e na criação de um ficheiro *bitstream* que realiza a configuração da FPGA. A configuração consiste na transferência do ficheiro *bitstream* para a FPGA.

A passagem da ferramenta XPS para a SDK é realizada através de um processo de exportação do projecto que consiste na criação de uma plataforma de *hardware* especificado por um ficheiro do tipo XML (*e**X**tensible **M**arkup **L**anguage)*, e na transferência do ficheiro *bitstream* para o SDK. Na ferramenta SDK pode-se realizar o desenvolvimento de aplicações utilizando as linguagens de programação C ou C++. A ferramenta SDK permite também a compilação e a depuração *(debug)* das aplicações. Caso a FPGA não tenha sido configurado na ferramenta XPS, a configuração do dispositivo pode ser realizada no SDK.

1.2.1.1 *Xilinx Platform Studio*

A ferramenta *Xilinx Platform Studio* apresenta um ambiente de desenvolvimento focado na componente de *hardware* de um sistema embebido. Para desenvolver um projecto de *hardware,* a ferramenta XPS disponibiliza uma aplicação designada de *Base System Builder* (BSB). A aplicação BSB proporciona uma forma expedita de desenvolver um sistema embebido completo e funcional para ser implementado numa FPGA da *Xilinx*, que recomenda a sua utilização para estabelecer as bases de um sistema embebido para a implementação nas placas equipadas com as suas FPGAs [3]. A aplicação BSB permite escolher a directoria onde os ficheiros do projecto ficarão guardados, permite selecionar a placa e a FPGA da implementação, selecionar e configurar um microprocessador e suas interfaces de Entrada/Saída, permite adicionar os periféricos pretendidos, e ainda produz um relatório resumido do sistema assim criado [3]. Após a conclusão do desenvolvimento da componente de *hardware*, são criados alguns ficheiros importantes para o projecto, descritos a seguir [3]:

- Ficheiro *system.xmp:* é o ficheiro de principal do projecto; a ferramenta XPS lê este ficheiro e apresenta graficamente o seu conteúdo através da interface do utilizador. Contém todas as informações sobre o projecto.

- Ficheiro *system.mhs:* representa as especificações de *hardware* (*MHS - Microprocessor Hardware Specification)*. Apresenta num formato de texto os elementos do sistema, os seus parâmetros e as suas ligações. Representa as bases de *hardware* do projeto. A descrição da plataforma de *hardware* é realizada neste ficheiro, constituindo a principal fonte que representa a componente de *hardware* do sistema embebido. A ferramenta XPS sintetiza este ficheiro em *netlists* que são enviadas para a FPGA.

- Ficheiro *system.mss*: representa as especificações de *software (MSS- Microprocessor Software Specification)*, descrevendo os elementos do sistema e os vários parâmetros de *software* associados aos periféricos num formato de texto.

Depois do desenvolvimento da componente de *hardware* do sistema, o projecto deve ser convertido num ficheiro binário chamado de *bitstream,* que é requerido para implementação na FPGA. Neste processo, a ferramenta XPS realiza três passos: primeiro gera um ficheiro *netlist* representativo da plataforma de *hardware*, no qual é desenvolvida uma representação HDL *(Hardware Description Language)* do ficheiro MHS, depois o

projecto é mapeado na lógica da FPGA, e finalmente é gerado o ficheiro *bitstream* que efectua a configuração da FPGA [3].

Após a conclusão da componente de *hardware* do sistema, este pode ser exportado para a ferramenta SDK, permitindo assim o desenvolvimento de aplicações de *software*. Ao exportar o projecto para a ferramenta SDK, esta fica com um ficheiro, em XML, que disponibiliza as especificações de *hardware.*

1.2.1.2 *Software Development Kit*

A ferramenta *Software Development Kit* permite o desenvolvimento de aplicações de *software* para sistemas embebidos com as linguagens de programação C e C++. A ferramenta SDK é uma ferramenta complementar à XPS.

No desenvolvimento de um projecto de *software*, cria-se uma plataforma de *hardware* que representa o sistema desenvolvido utilizando XPS; esta plataforma inclui o ficheiro XML e o ficheiro *bitstream*. Seguidamente escolhe-se um pacote de suporte, chamado *Board Support Package* que corresponde a um conjunto de bibliotecas de funções requeridas para as aplicações e de *drivers* para o sistema.

1.3 Sistemas Embebidos

Os sistemas embebidos são sistemas de processamento focados na realização de uma tarefa específica e predefinida. Incluem um microprocessador embebido, que está programado para realizar tarefas predefinidas, sendo muito utilizados em aplicações de controlo. Os sistemas embebidos estão presentes nos mais diversos dispositivos electrónicos, tais como: telemóveis, câmaras fotográficas digitais, impressoras, máquinas de digitalização de documentos, sistemas electrónico de automóveis, etc.

1.4 Tecnologia *Field Programmable Gate Arrays*

Uma FPGA é um circuito integrado que pode ser configurado de acordo com as aplicações do utilizador. Uma FPGA é constituída por milhares de células lógicas, quase todas idênticas entre si, organizados como blocos lógicos configuráveis e interligadas por interruptores programáveis. Cada célula da FPGA pode ser programada para realizar uma determinada função. A configuração de FPGAs é normalmente realizada a partir de

linguagens de descrição de *hardware,* normalmente conhecidas por HDLs, de entre as quais se destacam o VHDL e o *Verilog*.

1.5 Estrutura da Tese

Esta dissertação está estruturada em mais 5 capítulos para além deste. O capítulo 2 é dedicado ao calorímetro hadrónico *TileCal*, focando-se a electrónica de frontaria e o testador dos seus sistemas electrónicos, o MobiDICK. É feita a descrição dos principais componentes que o constituem e das suas funcionalidades principais.

O testador MobiDICK 4 vai comunicar com o utilizador através de uma interface *Ethernet* incorporada num sistema embebido implementado numa FPGA. O capítulo 3 é dedicado aos detalhes da implementação de duas interfaces *Ethernet* candidatas à implementação no MobiDICK 4.

Foi também realizada, no âmbito deste trabalho, a implementação de um algoritmo de *Cyclic Redundancy Check* (CRC) dedicado à verificação da integridade de dados enviados pela electrónica de frontaria do *TileCal* e recebidos pelo testador por via de ligações ópticas. O capítulo 4 é dedicado aos detalhes da implementação desse algoritmo.

No capítulo 5 são descritos os testes efectuados para a validação das interfaces *Ethernet* e do módulo CRC, são também apresentados os resultados obtidos nos respectivos testes.

Finalmente no capítulo 6 é dedicado às conclusões, considerações finais e recomendações para futuros trabalhos.

Para além dos referidos capítulos, este trabalho apresenta 2 apêndices, o primeiro dedicado a uma breve descrição da tecnologia de comunicação *Ethernet* e o segundo contendo resultados obtidos utilizando a aplicação *Iperf*, assim como alguns resultados do teste PING.

Capítulo 2 Calorímetro Hadrónico *TileCal* e o Testador MobiDICK 4

Este capítulo é dedicado ao *TileCal* da experiência ATLAS/CERN e ao respectivo testador de sistemas electrónicos, o MobiDICK 4. Para ser proporcionada uma melhor compreensão do testador, é efectuada primeiramente uma breve descrição da estrutura e do princípio de funcionamento do calorímetro hadrónico *TileCal* e da electrónica de frontaria *(Front-end)* nele instalado.

O testador MobiDICK 4 tem como base uma versão anterior, o MobiDICK 3, e portanto, neste capítulo, será descrita brevemente esta anterior versão. Ao contrário do MobiDICK 3, o *hardware* do novo testador centra-se no uso de sistemas reconfiguráveis, mas os princípios de funcionamento e de realização de testes são exactamente os mesmos, isto é, o testador MobiDICK 4 apresenta a funcionalidade do testador MobiDICK 3. Para além disso, o facto de o novo testador ser implementado com tecnologia recente de electrónica reconfigurável permite efectuar alterações no projecto sem a necessidade de refazer o *hardware* do sistema e simultaneamente minimiza o peso e a dimensão do testador, o que se traduz numa melhor mobilidade, essencial para as deslocações à caverna do detector ATLAS.

2.1 Calorímetro Hadrónico *TileCal*

2.1.1 Estrutura

Um calorímetro serve para medir a energia de partículas que o atravessam. O *TileCal* mede a energia das partículas resultantes das colisões ocorridas durantes as experiências do LHC. Geralmente, existem dois tipos de calorímetros: calorímetros de amostragem e calorímetros homogéneos. Os calorímetros de amostragem utilizam diferentes tipos de materiais como absorvedor e meio activo para a criação do sinal para a aquisição. Os calorímetros homogéneos utilizam todo o volume do material para a criação do sinal, e normalmente as funções de absorção e criação do sinal são combinadas num único material. O *TileCal* é o calorímetro central do detector ATLAS, e é um calorímetro de amostragem que utiliza aço como absorvedor e lâminas cintiladoras de plástico, conhecidas por *Tiles,* como meio activo. Apresenta uma estrutura cilíndrica, com um raio interno de 2,28 m e um raio externo de 4,23 m, é constituído por 3 blocos cilíndricos, um bloco central (denominado *Long Barrel*) de comprimento aproximado de 5,60 m e dois blocos laterais (denominados de *Extended Barrels*), cada um de 2,65 m de comprimento (figura 2.1).

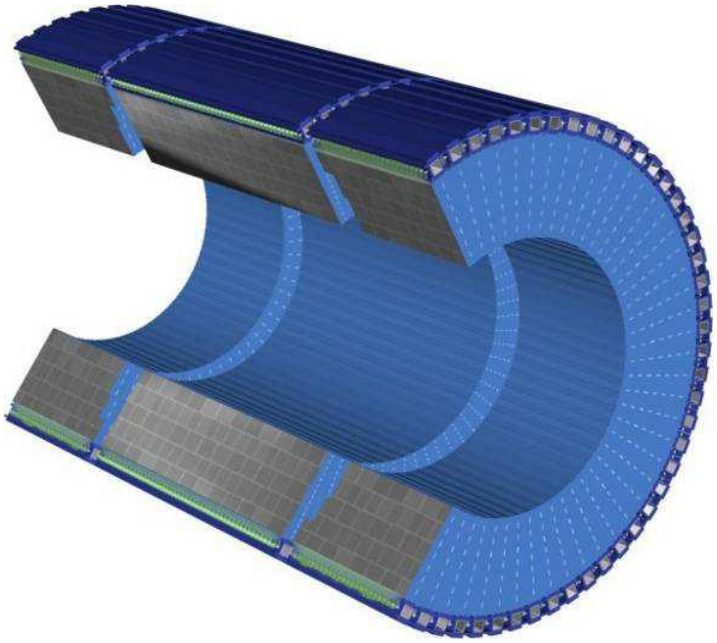


Figura 2.1: *Estrutura do calorímetro hadrónico TileCal* [1].

Cada bloco cilíndrico do *TileCal* é, por sua vez, dividido azimutalmente em 64 módulos. As lâminas cintiladoras são colocadas em planos perpendiculares ao feixe de partículas do LHC e inseridas em profundidade, o que permite uma boa homogeneidade na resolução em energia [1], figura 2.2.

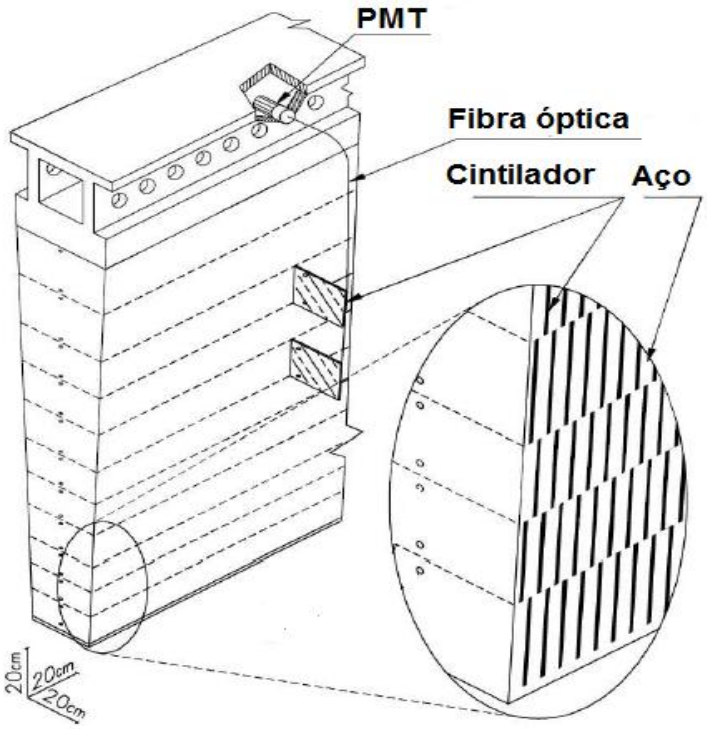


Figura 2.2: *Esquema da estrutura de amostragem do TileCal e integração dos cintiladores* [4].

2.1.2 Sistema de Aquisição

Durante as experiências do LHC, partículas ionizantes que atravessam o detector *TileCal* fazem com que fotões sejam produzidos nos cintiladores, sendo a intensidade desse fotões proporcional à energia depositada pelas respectivas partículas. Estes fotões são absorvidos e conduzidos por fibras ópticas de deslocamento de comprimento de onda *(WaveLength Shifting Fiber-WLSF)*, sendo a propagação da luz feita com base no fenómeno de reflexão total interna, até os fotões atingirem um conjunto de fotomultiplicadores (PMTs). Os fotomultiplicadores convertem os sinais luminosos em impulsos eléctricos, que servem de sinais de entrada da electrónica de frontaria, em concreto das placas *3in1.* Estas placas realizam o condicionamento e processamento dos sinais antes destes serem enviados ao sistema de digitalização. São utilizadas fibras ópticas do tipo WLSF porque o comprimento de onda da luz produzida pelos cintiladores encontra-se na região do Ultra-violeta e este tipo de fibra óptica desloca o comprimento de onda para um valor maior que é mais compatível com a sensibilidade do fotomultiplicador (o pico de sensibilidade ocorre para um comprimento de 420 nm) [2].

Os sinais analógicos recebidos pela electrónica de frontaria são digitalizados por um sistema de digitalização e subsequentemente enviados à electrónica de retaguarda *(Read-Out Drivers-ROD)* em pacotes de dados digitais.

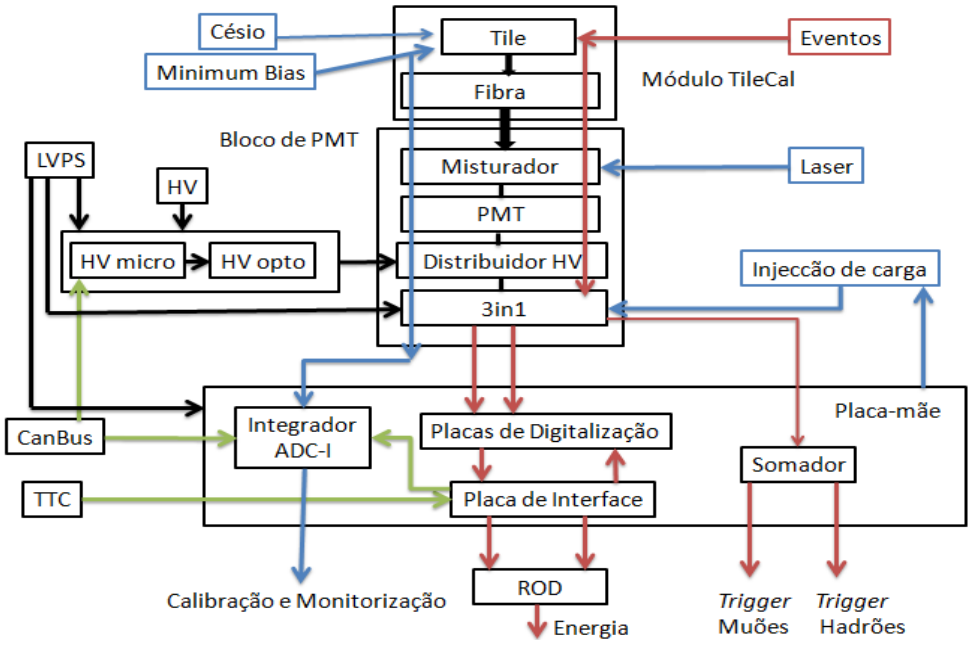


Figura 2.3: *Esquema do sistema de Aquisição e Leitura do TileCal* [5].

A figura 2.3 ilustra o esquema do sistema de aquisição e leitura de dados do *TileCal*. Este sistema dispõe de componentes que permitem a calibração e monitorização do calorímetro *TileCal*. Inclui um Sistema Laser integrado que permite o estudo, calibração e monitorização da resposta dos PMTs. O sistema "Césio" é utilizado para a verificação da qualidade e uniformidade do sistema óptico e opto-electrónico (cintiladores, fibras ópticas, PMTs) através da medição de respostas individuais de cada célula do *TileCal*. Este sistema tem como principal objectivo a determinação de uma referência absoluta para a escala de energia no *TileCal*. Os sinais produzidos pelos PMTs, em resposta aos fotões γ emitidos pela fonte de Césio, são medidos por um conversor-integrador (ADC-I) cuja função é digitalizar o integral desses sinais a uma frequência constante de 90 Hz [6]. Durante as experiências do LHC os eventos chamados *Minimum Bias* (resultantes de transferência de momentos relativamente pequenos que resultam das colisões protão-protão) são também adquiridos e utilizados para a monitorização do desempenho do *TileCal*.

O bloco HV *(High Voltage)* representa o sistema que fornece a alta tensão aos fotomultiplicadores. Para um certo conjunto de fotomultiplicadores há uma placa reguladora de alta tensão chamada de *HVopto*. Esta placa permite um ajustamento fino da alta tensão para cada fotomultiplicador, numa faixa de 350 V abaixo da tensão de entrada aplicada. Cada conjunto de duas placas *HVopto* é controlado por uma placa chamada *HVmicro*. A comunicação com a placa *HVmicro* é realizada através de uma interface *CanBus*. O sistema LVPS *(Low Voltage Power Supply)* representa o sistema que fornece baixas tensões de alimentação à electrónica de frontaria, nomeadamente às placas *3in1* e à Placa Principal.

2.2 Electrónica de Frontaria do *TileCal*

A electrónica de frontaria do *TileCal* encontra-se organizada em estruturas compactas denominadas de *Drawers.* Um conjunto de 2 *Drawers* inseridos numa barra de suporte forma um *Super-drawer*. Cada *Super-drawer* encontra-se associado a um módulo do calorímetro *TileCal*, sendo capaz de colectar informação vinda de 32 ou 45 canais do detector [5] (figura 2.4). Um *Super-drawer* do bloco cilíndrico central contém 45 fotomultiplicadores e um *Super-drawer* de cada um dos blocos laterais contém 32 fotomultiplicadores [2].

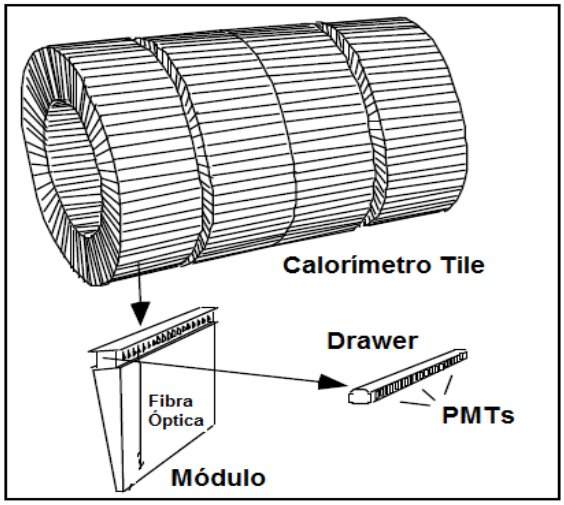


Figura 2.4: *Esquema do TileCal, ilustrando a disposição dos módulos e dos respectivos Drawers.*

A electrónica de frontaria do *Tilecal* é constituída pelos seguintes componentes principais: a Placa-Principal (*MotherBoard)*, o Bloco Fotomultiplicador, o Sistema de digitalização *(Digitizers)*, a Placa de Interface *(Interface Board)* [2, 7], o Somador Analógico e um Conversor Analógico-Digital chamado de ADC-I. A figura 2.5 ilustra a estrutura de um *Super-drawer* que suporta a electrónica de frontaria para a aquisição de sinais produzidos no calorímetro hadrónico *TileCal*.

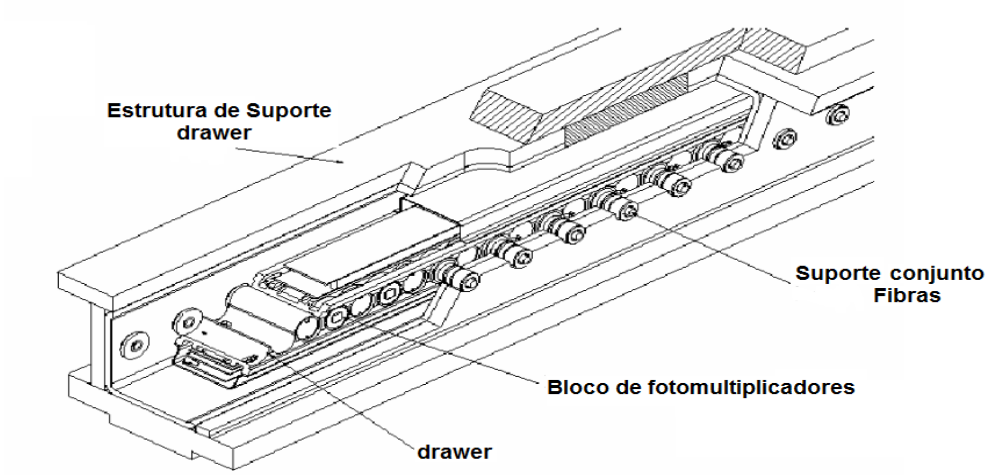


Figura 2.5: *Estrutura de um Super-drawer* [2] .

A seguir são descritos os principais componentes da electrónica de frontaria, assim como as suas funcionalidades.

2.2.1 Bloco Fotomultiplicador

A função de cada bloco fotomultiplicador é converter os fotões produzidos pelos cintiladores em sinais eléctricos. Cada bloco de fotomultiplicador é constituído por quatro componentes principais: um Tubo Fotomultiplicador (PMT), um Misturador de Luz *(Light Mixer)*, um Distribuidor de Alta Tensão *(HV divider)* e uma placa *3in1* [2, 7].

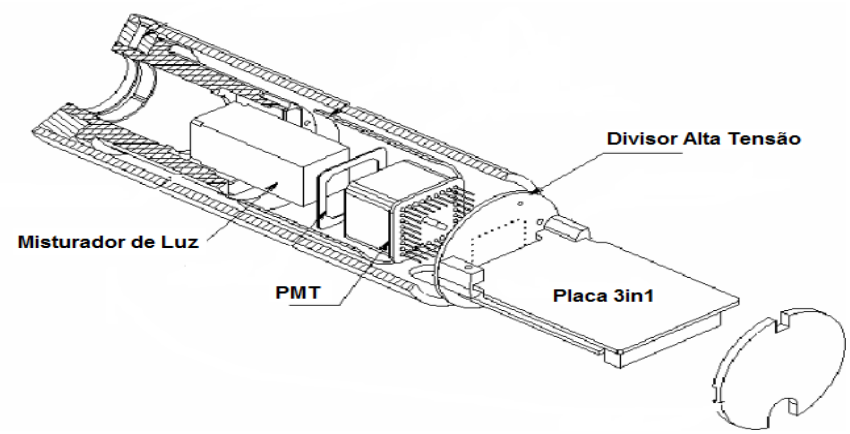


Figura 2.6: *Estrutura de um bloco de um fotomultiplicador* [2] .

A figura 2.6 ilustra a estrutura de um bloco de um fotomultiplicador. A seguir são descritos as funcionalidades de cada um deles:

Tubo Fotomultiplicador (PMT)

As conversões dos sinais luminosos (transmitidos através de fibras ópticas) para sinais eléctricos são realizadas por tubos fotomultiplicadores. Os fotomultiplicadores do modelo *Hamamatsu R7877* (figura 2.7) foram escolhidos, depois de serem realizados vários estudos que permitiram concluir que eles seriam o modelo mais adequado para integrar a electrónica de frontaria do *TileCal* [2, 7].

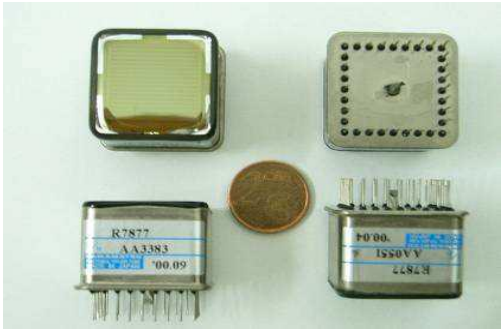


Figura 2.7: *Fotomultiplicador Hamamatsu R7877 utilizado na electrónica de frontaria do TileCal* [2].

Distribuidor de Alta Tensão

A principal função deste componente é distribuir a alta tensão entre os dínodos do fotomultiplicador, figura 2.8. Serve também como suporte de ligação permitindo que os fotomultiplicadores sejam ligados à electrónica de frontaria sem necessidade de cabos de interconexão adicionais, permitindo a redução da capacitância entre o fotomultiplicador e a restante electrónica de frontaria [2, 7].

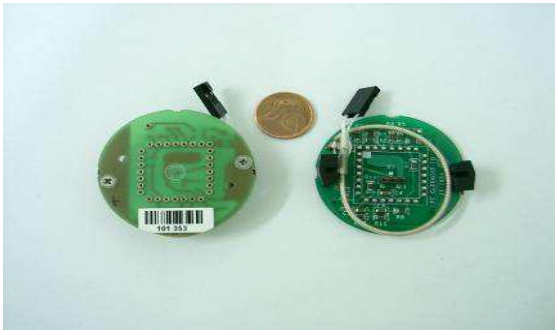


Figura 2.8: *Fotografia do divisor de alta tensão utilizado na electrónica de frontaria do TileCal* [2].

Misturador de Luz

O Misturador de Luz é posicionado entre as fibras ópticas e o fotocátodo do fotomultiplicador com o objectivo de optimizar a uniformidade de detecção. Permite uma mistura de sinais luminosos provenientes das diversas fibras ópticas evitando que exista uma correlação entre a posição de uma fibra óptica e a área do fotocátodo que recebe esses sinais luminosos [2, 7].

Placa 3in1

A placa *3in1* (figura 2.9) apresenta três funcionalidades [2]: condicionamento e processamento dos sinais dos PMTs por forma a constituírem os sinais de entrada do sistema de digitalização, calibração de injecção de carga e integração lenta dos sinais do fotomultiplicador para monitorização e calibração do sistema.

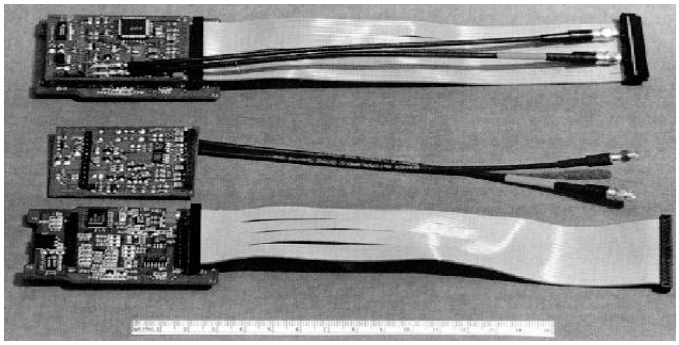


Figura 2.9*: Placa 3in1* [2] .

Cada placa *3in1* é constituída por um sistema de condicionamento e processamento de sinais (impulsos), um sistema de injecção de carga que permite a simulação de impulsos, um circuito integrador de carga para a calibração, um amplificador de ganho ajustável entre 0.5 e 32 e alguma lógica adicional de controlo [8]. O sistema de injecção de carga permite a calibração dos conversores Analógico-Digital do sistema de digitalização, permitindo a injecção directa de impulsos de carga de valores conhecidos. Estes impulsos são previamente digitalizados por um conjunto de conversores analógicos-digitais, a comparação entre os sinais digitalizados e sinais padrão permite determinar os factores de calibração [9].

O circuito integrador de carga é utilizado para a calibração e monitorização do desempenho do *Tilecal*. Durante as experiências do LHC, há colisões inelásticas p-p (protão-protão), com transferência de pequenos valores de momento, produzindo eventos chamados de *Minimum Bias*. Esses eventos, juntamente com o ruído intrínseco do *Tilecal*, constituem um dos limites ao desempenho do calorímetro. No entanto eles são utilizados para a monitorização da resposta do calorímetro, pois o circuito integrador de carga permite estudar a resposta do calorímetro através de uma integração lenta dos sinais recebidos, durante esses eventos. Esta monitorização é realizada *online* com o LHC em funcionamento, através da leitura dos sinais dos integradores de carga e sem interferir na aquisição normal de dados [9].

2.2.2 Sistema de Digitalização

O sistema de digitalização do *TileCal (TileCal Digitizer)* digitaliza os impulsos rápidos recebidos da placa *3in1* [2, 7]. O sistema consiste em duas cadeias de placas digitalizadoras ligadas à interface de leitura no centro dos *Super-drawers.* Cada placa digitalizadora realiza a discretização temporal dos sinais recebidos (com uma frequência de 40 MHz) e armazena dados provenientes de 6 fotomultiplicadores. Cada placa digitalizadora é constituída pelos seguintes componentes [2]:

- Dois conversores Analógico-Digital (ADC) de 12 *bits* de resolução por canal (num total de 48 canais).

- Dois dispositivos chamados de *TileDMU (Tile Data Management Unit)*, que são circuitos integrados ASIC *(Application Specific Integrated Circuit)* personalizados, com memórias concorrenciais *(pipeline)* para cada conjunto de três canais, com circuitos tampão *(buffers)* de leitura e funções de controlo.

- Um dispositivo chamado *TTCrx (Trigger Timing Control)* responsável pela temporização, programação e recepção de sinais de comandos enviados pela electrónica de retaguarda.

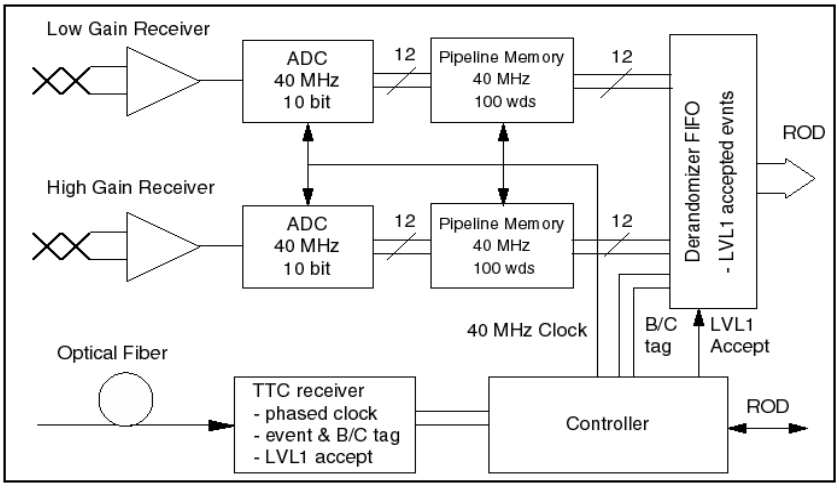


Figura 2.10: *Sistema de digitalização para um único canal.*

Conversores Analógico-Digitais

As placas *3in1* recebem os sinais analógicos dos fotomultiplicadores e realizam o seu condicionamento e processamento. Os sinais são posteriormente digitalizados. As placas *3in1* apresentam dois tipos de sinais de saída, sinais de baixa amplitude e sinais de amplitude elevada que são digitalizados pelos referidos conversores Analógico- Digitais de 12 *bits.*

Dispositivos *TileDMU*

Os dispositivos *TileDMU* são responsáveis pela formatação e ordenação dos dados digitalizados em pacotes de dados, e pelo envio desses pacotes de dados à Placa de Interface da electrónica de frontaria. Cada dispositivo *TileDMU* recebe e gere dados de 3 canais (correspondentes a sinais de 3 placas *3in1*). Cada placa digitalizadora dispõe de dois dispositivos *TileDMUs*. Existem 8 placas digitalizadoras por módulo do bloco cilíndrico central do *TileCal* contendo 48 fotomultiplicadores (48 canais) e 6 placas digitalizadoras por cada módulo dos blocos laterais contendo 36 fotomultiplicadores (36 canais possíveis, sendo necessários apenas 32) [7].

Tal como já foi referido, um *TileDMU* recebe dados de 3 canais que são armazenados em memórias *pipeline*. O sistema de digitalização só disponibiliza a leitura dos pacotes de dados após receber o comando *Trigger Nivel-1* (L1A) enviado pelo sistema TTC da electrónica de retaguarda. Esses pacotes de dados encontram-se armazenados localmente em memórias RAM de dois portos *(Buffers Derandomizers).* O comando L1A permite a sua transmissão para a Placa de Interface, que por sua vez os envia aos RODs *(Read-Out Drivers)*.

Dispositivo *TTCrx*

Para sincronizar os detectores com o feixe de partículas do LHC, é utilizado um sistema de temporização, decisão e controlo na electrónica de retaguarda, denominado de TTC *(Timing, Trigger and Control)*. Este sistema, sincronizado com o feixe do LHC, fornece o relógio à electrónica de frontaria. Apresenta várias funções que permitem controlar o sistema *TTCrx* do sistema de digitalização através do envio de comandos: por exemplo, a geração e envio do comando L1A é realizada pelo TTC. O sistema *TTCrx* é responsável pela descodificação do sinal gerado pelo TTC e pelos três relógios de 40 MHz das placas digitalizadoras [10].

2.2.3 Placa-Principal

A Placa-Principal recebe mensagens de controlo da electrónica de retaguarda via TTC, comanda a configuração das placas *3in1* (inicia a injecção de carga, controla o ganho do integrador) e controla também o somador analógico (2.3.5) permitindo activar/desactivar o sinal de saída desse somador analógico. Utiliza um canal *CanBus* que ao permitir a leitura das definições de controlo aplicada às placas *3in1*, permite aceder ao estado do sistema [4].

Os sinais de controlo podem ser enviados às placas *3in1* individualmente ou para todas as placas *3in1* simultaneamente. O sistema TTC é o sistema principal de controlo e permite a temporização correcta das funções associadas aos comandos que envia [4].

2.2.4 Placa de Interface

No calorímetro *TileCal* existe uma Placa de Interface por cada *Super-drawer* que apresenta duas funcionalidades principais [7]:

- Receber os sinais do sistema TTC e enviá-los ao sistema de digitalização que dispõe do dispositivo *TTCrx* para sua recepção.

- Receber os dados enviados pelas 6 ou 8 placas digitalizadoras de um *Super-drawer*, deserializá-los e enviá-los, por via de ligações ópticas, ao andar de entrada do sistema de leitura da electrónica de retaguarda, o ROD.

As ligações ópticas da Placa de Interface usam o protocolo *S-Link* assente em dispositivos *G-Link* como camada física. É usada uma placa de ligação *G-Link* equipada com um dispositivo transmissor HDMP1032 de 16 *bits* [11], operando a uma taxa de 640 Mbps, e a uma frequência de 40 MHz [7].

2.2.5 Somador Analógico

Este bloco recebe na entrada sinais analógicos provenientes de placas *3in1* e efectua a sua soma. Cada placa *3in1* é ligada a um destes blocos, podendo ser conectadas 6 placas *3in1* ao mesmo somador analógico. Existem 9 Somadores Analógicos em cada *Super-drawer* do bloco cilíndrico central e 7 em cada *Super-drawer* dos blocos laterais. Estes Somadores Analógicos apresentam como sinal de saída um único sinal analógico que serve de sinal de *Trigger* de nível 1 do detector ATLAS [8].

2.2.6 Conversor ADC-I

Corresponde a um conversor Analógico-Digital utilizado para digitalizar as correntes de calibração das placas *3in1*, isto é, os sinais de saída dos circuitos integradores de carga [8].

2.3 Testador de Sistemas Electrónicos do *TileCal* - MobiDICK

O testador MobiDICK *(Mobile Drawer Checking Integrity System)* é um sistema móvel de teste e verificação da integridade dos *Super-drawers* que suportam a electrónica de frontaria, depois de estes serem inseridos nos módulos do *TileCal*. Os primeiros *Super-drawers* foram montados e testados pelo LPC-CF *(Laboratoire de Physique Corpusculaire – Clermont Ferrand)* em França e posteriormente transportados para o CERN. Para facilitar o transporte, os *Super-drawer* foram divididos em dois *Drawers* sendo a montagem final e a inserção nos módulos realizada no CERN. A integridade dos *Super-drawers* e a ligação entre os dois *Drawers* necessitou de ser verificada no CERN, antes da sua inserção nos módulos do *TileCal*. Assim, foi implementado o primeiro sistema MobiDICK para efectuar a sua verificação [12]. Para além disso, o MobiDICK também pode ser utilizado para diagnosticar problemas nos *Drawers* já instalados na caverna do ATLAS, daí a necessidade de ser um sistema móvel. A primeira versão deste testador foi implementada em 2003, para os testes dos primeiros *Super-drawers*. Desde então foram realizadas duas actualizações (MobiDICK 2 e MobiDICK 3). A versão 4, em desenvolvimento baseia-se na versão 3 e conta com a colaboração e participação de várias instituições e grupos de investigação de diferentes países. A implementação e teste do Mobidick 4 são realizados nos laboratórios do CERN.

2.3.1 Testador MobiDICK 3

O MobiDICK 3, permite testar todas as funcionalidades de um *Super-drawer*. Assim, a nova versão usa esta versão como base (figura 2.11). Na versão 3, o MobiDICK encontra-se dentro de uma caixa de alumínio com dimensões 50x41x33 cm, e um peso aproximado de 20 kg [8].

Figura 2.11: *Fotografia do MobiDICK 3, a versão actual.*

2.3.1.1 O *Hardware* do MobiDICK 3

O *hardware* encontra-se numa *crate* VME *(Versa Module Europa)* contendo um conjunto de 5 placas VME, conectadas por um barramento VME, sendo controlado por um processador [8]:

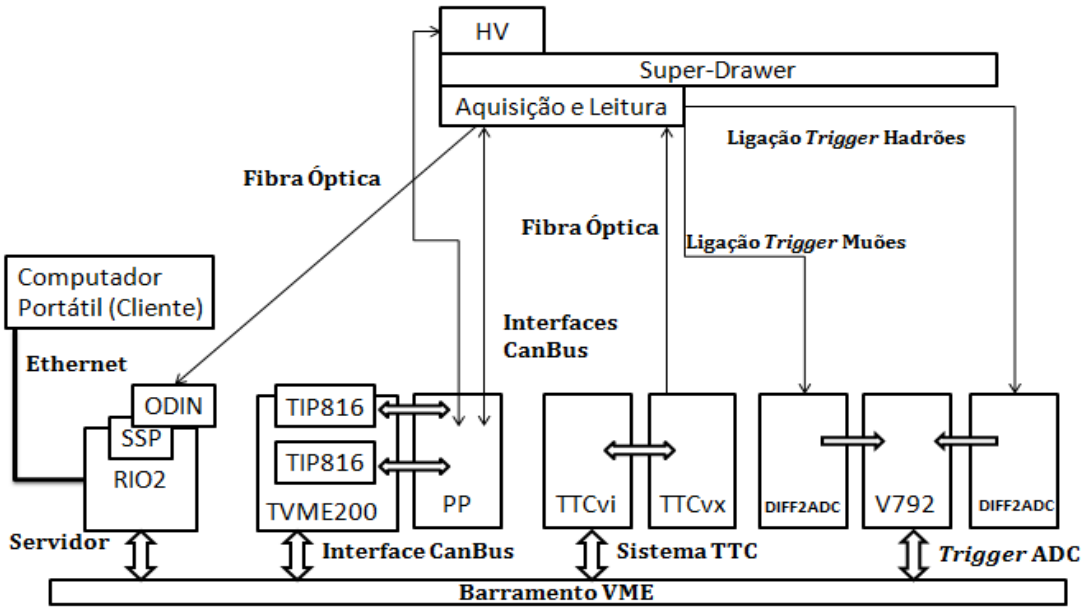


Figura 2.12: *Esquema da arquitectura do MobiDICK 3* [8].

A figura 2.12 ilustra o *hardware* do MobiDICK 3, sendo constituído pelos seguintes componentes:

Processador (Servidor)

Consiste numa placa VME contendo o processador RIO2 que funciona como servidor do MobiDICK 3. Este é responsável pela realização de testes, pela comunicação com o utilizador e pelo controlo de todas as outras placas VME. Encontra-se equipada com uma interface chamada de SSP *(Simple S-Link to PMC)* onde está implementada uma interface ODIN *(Optical Dual G-Link S-Link).* A interface ODIN é conectada à Placa de Interface da electrónica de frontaria por fibras ópticas e permite a recepção de pacotes de dados.

Interface *CanBus*

Consiste também numa placa VME TVME200, com duas interfaces *CanBus* independentes implementadas por módulos TIP816 [13]. Estes dois módulos encontram-se conectados a uma placa projectada e implementada pelo LPC-CF, chamada de PP que é utilizada para fornecer uma tensão de alimentação de 12 V à interface *CanBus* do *Super-drawer* e para enviar os sinais dos dois módulos TIP816 através de um único cabo para o respectivo *Super-drawer*.

Sistema TTC

É constituído por 2 módulos: uma placa VME *TTCvi* e uma placa *TTCvx*. A placa *TTCvi* gera comandos (sinais eléctricos) para a electrónica de frontaria, e encontra-se conectada à placa *TTCvx* que é responsável pela conversão dos sinais eléctricos em sinais ópticos que são enviados para a electrónica de frontaria através de uma fibra óptica.

Placa *Trigger ADC*

É constituída por 3 módulos. Um módulo CAEN V792 [14] digitaliza os sinais enviados pelos Somadores Analógicos dos *Super-drawers.* Estes sinais são diferenciais, sendo por isso processados por duas placas personalizadas chamadas de DIFF2ADC, uma para o cabo do *Trigger* de Muões e outra para o cabo do *Trigger* de Hadrões.

Placa de alimentação HV *(High Voltage)* e Placa *LED Driver*

A placa de alimentação HV é composta por dois módulos. Um deles é uma placa personalizada chamada de *HVLED* que fornece uma alta tensão de -830 V aos *Super-drawers*. O segundo módulo consiste numa placa personalizada chamada de *LED Driver* que fornece uma tensão de 20 V, que permite o funcionamento de LEDs *(Light Emission Diodes)* no modo contínuo

ou pulsado (impulsos de duração igual a 20 ns). Esta placa é controlada pela placa DIFF2ADC que apresenta interfaces Entrada/Saída, acessíveis através do barramento VME.

2.3.1.2 O *Software* do MobiDICK 3

A componente de *software* do MobiDICK 3 divide-se em duas partes: uma delas encontra-se no processador (na *crate* VME) e a outra parte encontra-se no computador portátil ligado ao MobiDICK. Comunicam-se pelo protocolo TCP/IP num modelo Cliente/Servidor, tendo como suporte a tecnologia *Ethernet*.

Cliente

O *software* no computador portátil, chamado *Willy*, funciona como cliente permitindo a interface entre o operador e o sistema *TileCal*. É uma aplicação desenvolvida em C++, em ambiente Linux, utilizando as bibliotecas do ROOT. O *Willy* solicita ao servidor a realização de testes electrónicos e o envio de resultados, que são analisados pela aplicação Cliente permitindo que o operador avalie os problemas nos *Super-drawers*, caso existam [8].

Servidor

O servidor é uma aplicação que recebe os pedidos de realização de testes enviados pela aplicação *Willy* e envia os resultados. Este servidor é uma aplicação desenvolvida em C, funcionando em ambiente *LynxOS*, o sistema operativo instalado no processador RIO2 [8].

2.3.2 O Testador MobiDICK 4

Tal como já foi referido, o MobiDICK 4 terá por base o MobiDICK 3, já descrito na secção anterior. O desenvolvimento de uma nova versão do MobiDICK foi motivado pelas seguintes razões:

- Actualmente existem 3 testadores, e com 4 sistemas disponíveis será possível testar 4 blocos cilíndricos do *TileCal* em simultâneo.
- Desenvolver um sistema a partir de tecnologias actuais de fácil actualização e manutenção (reconfiguráveis), substituindo as tecnologias já obsoletas.
- Reduzir o tamanho e o peso, permitindo uma melhoria da mobilidade.
- Ficar com um sistema compatível com a nova electrónica de frontaria.

2.3.2.1 Arquitectura

Na figura 2.13 encontra-se representada a arquitectura do MobiDICK 4. Ao contrário do MobiDICK 3, que utiliza placas VME, aquele sistema é baseado em sistemas reconfiguráveis. No núcleo do MobiDICK 4 há uma placa *Xilinx* ML507 equipada com uma FPGA Virtex-5, que efectua a substituição das placas VME por módulos VHDL implementados na FPGA, e por algumas placas electrónicas proprietárias.

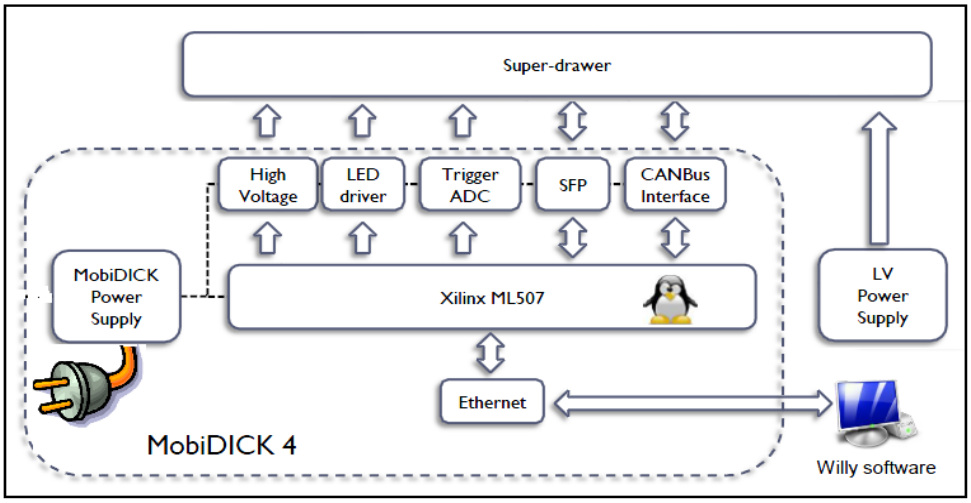


Figura 2.13: *Arquitectura do MobiDICK 4.*

A tabela 2.1 mostra a correspondência entre os dois sistemas e as respectivas funções. A placa de desenvolvimento ML507 permite que sejam implementados em paralelo vários módulos na FPGA, que no sistema anterior correspondiam a placas VME individuais.

Funcionalidade	**MobiDICK 3**	**MobiDICK 4**
Processador: Servidor com *LynxOS*	RIO2	ML507 (virtex-5) + Módulo óptico SFP + Conversores Série ↔ *Canbus*
Fornece o sinal L1A e os comandos TTC	TTCvi	
	TTCvx	
Recebe dados da Placa de Interface	SSP+ODIN	
Comunicação *Canbus*	TVME200	
	TIP816	
Digitaliza os sinais dos cabos do *trigger* de muões e hadrões	DIFF2ADC	Placa *ADC Trigger*
	CAEN V792	
HV fornece a alta tensão aos PMTs LED Driver para o sistema de calibração	Placa HVLED	Nova Placa HVED
Baixa tensão de alimentação da electrónica de frontaria	Placa *LV Power supply*	Placa *LV Power supply*

Tabela 2.1: *Correspondência entre a versão actual e a nova versão do MobiDICK.*

O desenvolvimento do testador MobiDICK 4 está a ser realizado por vários grupos de investigação de diferentes países. O projecto foi dividido em partes e em tarefas, distribuídas aos vários grupos. Cada grupo é responsável pela implementação e respectivo teste da componente que lhe foi atribuída. A tarefa do LIP, que deu origem ao trabalho aqui apresentado, focou a implementação e testes de interfaces *Ethernet* disponíveis para a implementação num sistema embebido e pela implementação em VHDL na placa ML507 de um algoritmo de verificação da integridade de dados enviados pela electrónica de frontaria e recebidos pelo MobiDICK 4. A seguir é apresentada a descrição de todos os componentes do MobiDICK 4:

Placa de Controlo

A placa de controlo do MobiDICK 4 é uma placa ML507 equipada com uma FPGA virtex-5 da *Xilinx*, figura 2.14.

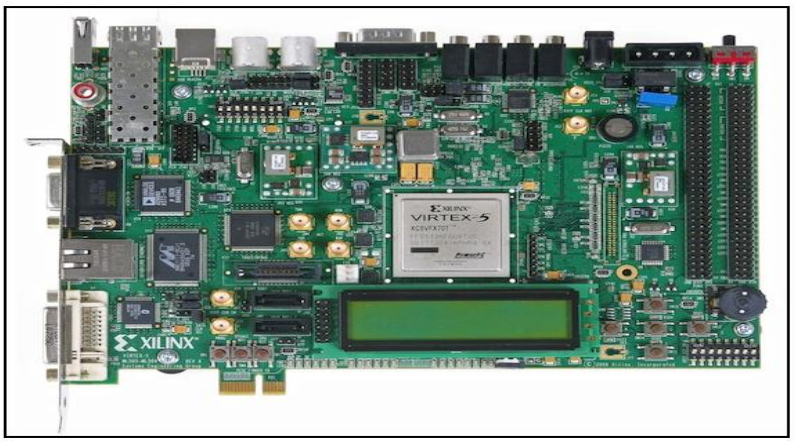


Figura 2.14: *Placa ML507 equipado com uma FPGA Virtex 5 da Xilinx.*

Esta placa encontra-se equipada com um microprocessador embebido, o *PowerPC 440*, 256 MBytes de memória RAM, e interfaces de comunicação tais como *Ethernet*, USB, COM assim como um módulo óptico SFP *(Small Form Pluggable)* que permite ligações através de fibras ópticas. Esta placa permite o controlo de todos os dispositivos referidos.

Módulo SFP

Este módulo representa um conector óptico que permite a ligação por fibras ópticas à Placa de Interface da electrónica de frontaria, por onde se faz o envio de comandos L1A e recepção de dados enviados pelos *Super-drawers*.

Placa *Low Voltage Power Supply*

É uma placa de alimentação comercial que alimenta os *Super-drawers*.

Placa *High Voltage*

A placa *High Voltage* fornece a alta tensão aos fotomultiplicadores do *TileCal* durante a realização de testes. A figura 2.15 é uma fotografia de um protótipo desta placa.

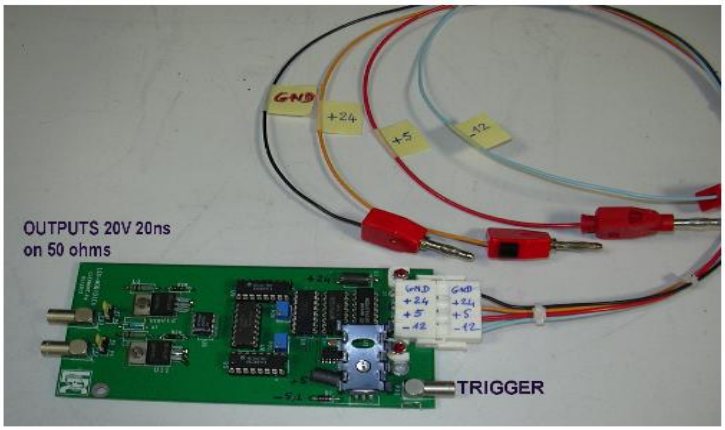


Figura 2.15: *Fotografia de um protótipo da placa High Voltage do MobiDICK 4.*

Placa *LED Driver*

Esta placa permite testar e estudar a resposta dos *Super-drawers* a impulsos luminosos. Gera impulsos eléctricos que acendem dois LEDs em modo pulsado, obtendo-se assim, os sinais luminosos para os *Super-drawers.* A figura 2.16 é uma fotografia de um protótipo desta placa.

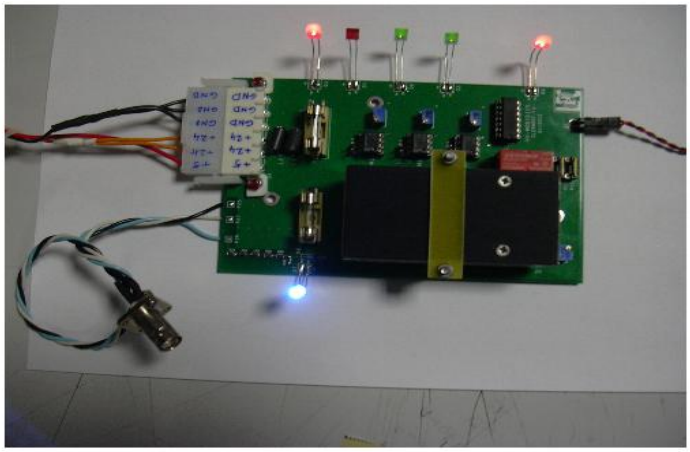


Figura 2.16: *Protótipo da placa LED Driver do MobiDICK 4.*

Placa *Trigger* ADC

Esta placa é responsável pela digitalização dos sinais analógicos provenientes dos *Super-drawers* através dos cabos do *Trigger* de Muões/Hadrões do *TileCal*. É constituída por dois conversores Analógico-Digitais ADS5271 da *Texas Instruments* com 12 *bits* de resolução, 8 entradas diferenciais e uma taxa de amostragem de 40 MSps. Os sinais digitais são serializados e

enviados à Placa de Controlo ML507 via um conector de expansão a uma taxa de 640 Mbps. A figura 2.17 representa um protótipo desta placa.

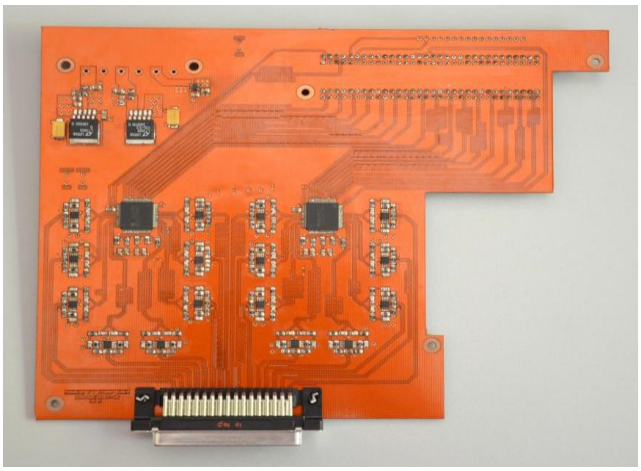


Figura 2.17: *Protótipo da placa Trigger ADC.*

Interfaces *CanBus*

O testador utiliza duas intefaces *CanBus* que permitem a comunicação com a placa HV e o ADC integrador (ADC-I) dos *Super-drawer*. Como na Placa de Controlo há portas RS-232, são utilizados Conversores *Série* ↔ *CanBus* comerciais que implementam o protocolo CAN. A figura 2.18 representa um dos Conversores *Série* ↔ *CanBus* utilizados no MobiDICK 4.

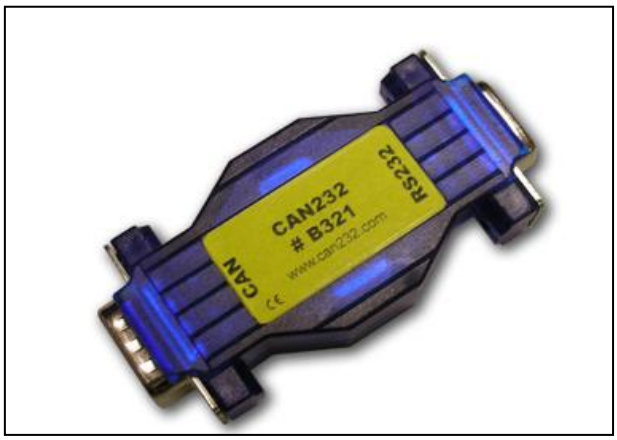


Figura 2.18: *Conversor Série↔CanBus comercial utilizado no MobiDICK 4.*

Interface *Ethernet*

O sistema comunica com o utilizador através da tecnologia *Ethernet* com recurso ao protocolo TCP/IP. A placa de controlo ML507 dispõe de *Ethernet* (sendo necessária a implementação de um controlador MAC) que é utilizada para a comunicação com o utilizador.

Placa *Power Supply*

Esta placa (figura 2.19) fornece e distribui a tensão de alimentação às diferentes placas do MobiDICK 4 e converte o sinal AC de 220 V para os diferentes níveis de tensão DC requeridos para o sistema.

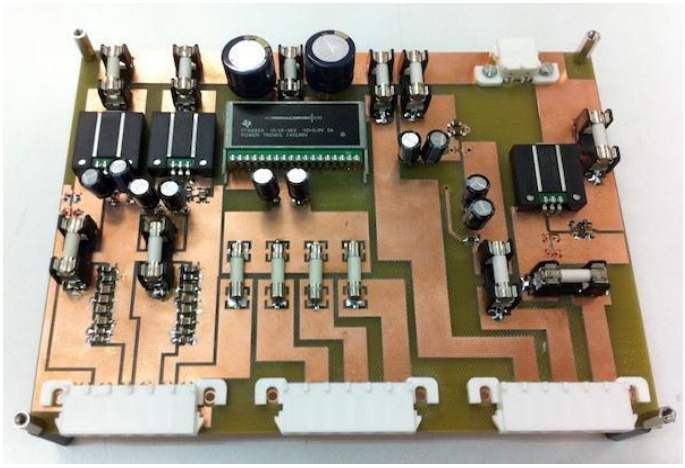


Figura 2.19: *Protótipo da Placa Power Supply do MobiDICK 4.*

Aplicação *Willy*

Willy é a aplicação de comunicação, via *Ethernet*, entre o utilizador e o sistema MobiDICK (placa ML507). O *Willy* apresenta a funcionalidade necessária à realização de testes e à recepção dos resultados. Está instalado num PC portátil sobre *Linux*.

2.3.2.2 Sistema Embebido do MobiDICK 4

Na Placa de Controlo ML507 equipada com a FPGA Virtex-5 está implementado um sistema *Linux (Linux kernel 2.6.29)*, executado pelo microprocessador embebido *PowerPC 440* operando a 400 MHz, que permite o controlo de todos os dispositivos externos, figura 2.20. Este sistema apresenta três funções principais:

- Gerir os dados recebidos através do módulo SFP e da placa *Trigger ADC.*
- Controlar e gerir o funcionamento dos Conversores *Série* ↔*CanBus*.
- Receber pedidos de realização por parte do *Willy*, e reenviar os resultados obtidos via *Ethernet*. Nesse sistema embebido é implementado o *software* servidor *Glue SW*, que recebe os pedidos de testes e trata da sua execução.

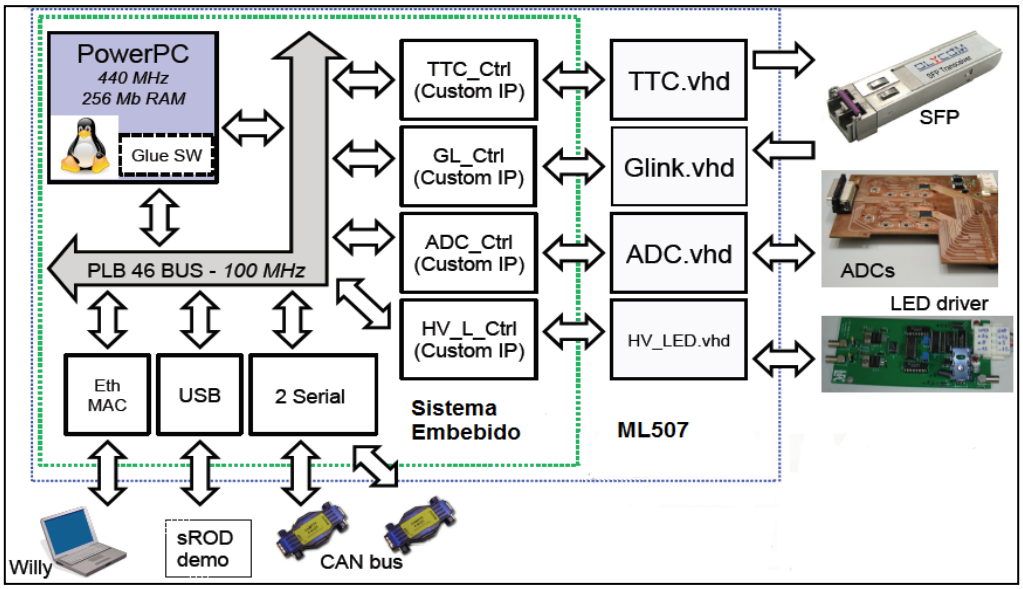


Figura 2.20: *Sistema embebido do MobiDICK 4.*

Neste sistema, o microprocessador *PowerPC 440* é o módulo central que controla os restantes módulos através do barramento *Processor Local Bus* (PLB) que será descrito mais adiante. Neste sistema são implementados 4 módulos personalizados: *TTC_Ctrl, GL_Ctrl, ADC_Ctrl* e *HV_L_Ctrl*. Dispõe de uma interface *Ethernet* (TMAC) para comunicar com a aplicação *Willy* do computador portátil, de duas interfaces de comunicação série para ligação aos Conversores *Série* ↔ *CanBus*, e de uma interface USB que será utilizada futuramente para a comunicação com um sistema de teste da nova electrónica de frontaria chamado provisoriamente *Demonstrator*. A seguir são descritas as funcionalidades oferecidas pelos componentes deste sistema embebido:

Microprocessador

O microprocessador embebido implementado é o *PowerPC 440*, com um relógio de 440 MHz, que controla todos os outros módulos através do barramento *Processor Local Bus* (PLB), que opera a uma frequência de 100 MHz. Foi instalada o sistema operativo *Linux* neste microprocessador. O servidor responsável pela realização de testes electrónicos é executado pelo microprocessador embebido *PowerPC 440*.

Interface *Ethernet MAC*

A partir dos resultados apresentados neste trabalho, concluiu-se que a interface MAC recomendada para esta implementação seria a interface *Tri-Mode Media Access Controller* (TMAC). Esta interface permite uma comunicação mais eficiente (mais rápida, menos erros) com a aplicação *Willy* no computador do que a interface *Ethernet Lite Media Access Controller*, que também era uma candidata à implementação.

Interface USB

O sistema embebido tem uma interface USB *(Universal Serial Bus)* que permite que o MobiDICK 4 seja compatível com o *Demonstrator*, um sistema actualmente em desenvolvimento que fará o teste da nova electrónica de frontaria do *TileCal*. Esta interface USB vai permitir a comunicação entre o MobiDICK 4 e o *Demonstrator*.

Interfaces *Serial*

Duas interfaces de comunicação série permitem que as portas RS-232 da placa ML507 sejam conectados a dois conversores *Série* ↔ *CanBus* utilizados na comunicação com os *Super-drawers* do TileCal.

TTC.vhd

Este módulo desenvolvido em VHDL, representa um emulador do sistema TTC da electrónica de retaguarda do *TileCal*. Permite o envio de comandos aos *Super-drawers* e configurar o módulo *TTCrx* do sistema de digitalização da electrónica de frontaria. Esses comandos são enviados através de ligações ópticas via módulo SFP.

TTC_Ctrl

É um núcleo proprietário *(Intellectual Property Core)* que permite o controlo do módulo *TTC.vhd* pelo microprocessador. É também uma interface entre o módulo *TTC.vhd* ao barramento PLB permitindo a comunicação com o microprocessador.

GLink.vhd

Este módulo, desenvolvido em VHDL, é um emulador do receptor *G-Link* (HDMP-1024) da electrónica de retaguarda. Na electrónica de retaguarda é implementado um circuito integrado receptor HDMP-1024, capaz de suportar comunicações com velocidades elevadas. O componente HDMP-1024 é utilizado para receber e de-serializar dados serializados enviados pelo componente transmissor HDMP-1032 da electrónica de frontaria, através de ligações ópticas, utilizando o protocolo *S-Link*. Este módulo recebe, pelo SFP, os dados enviados pela Placa de Interface da electrónica de frontaria. Inclui também um algoritmo de verificação da integridade dos dados enviados pela Placa de Interface denominado *Cyclic Redundancy Check* (CRC).

GL_Ctrl

É um núcleo proprietário que permite o controlo do módulo *GLink.vhd* por parte do microprocessador. É também uma interface de *GLink.vhd* ao barramento PLB, permitindo estabelecer a comunicação com o microprocessador.

ADC.vhd

É um módulo desenvolvido em VHDL para a recepção de dados digitalizados pela placa *Trigger* ADC. A placa *ADC Trigger* recebe sinais analógicos enviados da electrónica de frontaria, digitaliza-os e envia-os à placa ML507, que dispõe deste módulo para a sua recepção.

ADC_Crtl

É um núcleo proprietário que fornece uma interface do módulo *ADC.vhd* ao barramento PLB, permitindo o seu controlo através do microprocessador.

HV_LED.vhd

Um módulo, desenvolvido em VHDL, que permite a interação, por parte da placa ML507, com as placas *HV e LED Driver.*

HV_L_Ctrl

Um núcleo proprietário para o controlo do módulo *HV_LED.vhd*, que fornece uma interface ao barramento PLB, permitindo a comunicação com o microprocessador.

2.3.2.3 O *Software* do MobiDICK 4

O *software* do MobiDICK 4, assim como o do Mobidick 3, está dividido em duas partes:

- **Cliente *(Willy)*:** apresenta uma interface gráfica ao operador, envia comandos e pedidos de testes ao servidor.
- **Servidor:** Controla os componentes de *hardware*, realiza os testes e envia os resultados ao Cliente. É implementado em C++.

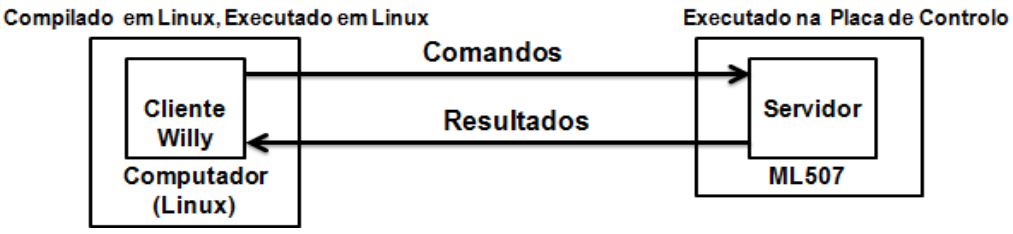


Figura 2.21: *Componente de software do MobiDICK 4.*

As duas partes, Cliente e Servidor comunicam-se através de uma interface *Ethernet* com recurso ao protocolo TCP/IP. O Cliente envia pedidos para a realização de testes, o servidor recebe-os e realiza os testes, analisa os resultados, e envia-os ao Cliente (figura 2.21).

2.3.3 Testes Digitais Efectuados pelo MobiDICK

Os testes dos *Super-drawers* do *TileCal* são realizados por partes, havendo um conjunto de testes realizados pelo MobiDICK. Esses testes são: *CommMB, Adder, DigShape, DigNoise, Integ, CommHV, Opto, NominalHV, IntegHV, DigShapeLED*. Os resultados do teste de cada *Super-drawer* são guardados e podem ser analisados posteriormente. Seguidamente é apresentada uma breve descrição de todos esses testes [8].

2.3.3.1 Teste *CommMB*

É o primeiro e o mais simples que se efectua a um *Super-drawer.* É realizado com o objectivo de verificar a comunicação com a placa ADC-I, com as placas *3in1* e com os dispositivos *TTCrx*. Consiste das seguintes etapas:

1. É estabelecida a comunicação com a placa ADC-I através da interface *CanBus*. É enviado um comando *IDALLOC DEFID*. Se a comunicação for bem sucedida, é enviado como resposta o Número de Série da respectiva placa ADC-I.

2. Utilizando o Número de Série da placa ADC-I (que é um parâmetro único para cada *Super-drawer*) é identificado o *Super-drawer* em teste e extraídos os seus parâmetros da base de dados de *Super-drawers*.

3. É verificada a versão do *Firmware* da placa ADC-I.

4. É verificada a comunicação com as placas *3in1* através da placa ADC-I. Todos os *bits* de controlo são invertidos (para teste) e o registo é lido para verificar se os comandos foram enviados com sucesso. Essas operações são realizadas via *CanBus*.

5. É verificada a comunicação com cada placa *3in1* através do *TTCrx*. Todos os *bits* de controlo dessas placas são invertidos, através de comandos enviados pelo sistema TTC. O registo de estado é lido através da interface *CanBus*, com o objectivo de verificar se os comandos foram executados com sucesso.

6. O endereço do dispositivo *TTCrx* é verificado: a primeira placa *3in1* a funcionar correctamente é selecionada e verificada três vezes com os testes anteriores, utilizando o

endereço de *TTCrx* esperado. O endereço esperado é subtraído e adicionado de uma unidade. Utilizando estes dois últimos endereços alterados, a comunicação com a placa 3in1 deverá falhar.

2.3.3.2 Teste *Adder*

Este teste tem por objectivo a verificação dos Somadores Analógicos utilizando o circuito de injecção de carga. Para verificar se não há falhas nos Somadores Analógicos, sinais de saída provenientes de 6 placas *3in1*, correspondendo a cargas eléctricas de valores conhecidos são injectados um por um nos somadores, permitindo verificar em cada passo se a variação no sinal de saída do somador é consistente com a esperada. Todos os comandos são enviados pelo sistema TTC, seguindo as seguintes etapas.

1. Todas as placas *3in1* são configuradas para que a entrada do circuito de injecção de carga seja desactivada. Depois, os sinais provenientes das ligações aos *Triggers* são digitalizados utilizando a placa *Trigger ADC*. Esses dados digitalizados são tomados como os valores de referência *(pedestals)* para cada Somador Analógico.

2. A entrada do circuito de injecção de carga é activada apenas numa das placas *3in1*, sendo injectada um sinal de carga igual a 7 pC.

3. Os sinais provenientes das ligações aos *Triggers* são novamente digitalizados e comparados com os valores de referência do passo 1. As diferenças entre eles são comparadas com os valores esperados. Espera-se que não haja variação no sinal de saída dos somadores que não estão ligados à placa *3in1* em questão, ao passo que se espera que haja um aumento de valor conhecido, na saída do somador que se encontra ligado à placa *3in1*, sob teste. Estes sinais digitalizados tornam-se os novos valores de referência.

4. O circuito de injecção de carga é activado em mais uma das placas *3in1*, regressando-se à etapa 3. Este processo é realizado para todas as placas *3in1*.

2.3.3.3 Teste *DigShape*

Este teste verifica a funcionalidade do sistema de digitalização e a aquisição de dados, utilizando o circuito de injecção de carga. As Placas de Controlo e todas as Placas Digitalizadoras são primeiramente configuradas. As placas *3in1* são configuradas activando o circuito de injecção de carga. São realizados dois passos:

1. É injectado um sinal de carga pequena (10 pC) para testar os circuitos de ganho elevado. Este sinal é injectado em todas as placas *3in1* simultaneamente, permitido a detecção de falhas no sistema de digitalização muito rapidamente.

2. Desta vez é injectado um sinal de carga maior (800 pC) para testar os circuitos de ganho reduzido. Este sinal é injectado apenas numa única placa *3in1*, para verificar a conexão com as placas de digitalização, permitindo verificar a placa *3in1* que poderá não estar conectada ao respectivo canal do sistema de digitalização.

Todos os comandos são enviados através do sistema TTC.

2.3.3.4 Testes *DigNoise* e *DigNoiseHV*

Estes dois testes têm como finalidade a medição do ruído das placas de digitalização e a verificação da integridade de dados, utilizando as seguintes etapas:

1. São verificados os BCID *(Bunch Crossing IDentifier)* de todos os *TileDMUs (Tile Data Management Unit)*.

2. Todos cálculos de CRC são realizados e os seus valores verificados.

3. É calculado e verificado o valor RMS *(Root Mean Square)* do ruído das placas de digitalização para cada canal.

Estes testes são realizados duas vezes, nas seguintes condições:

1. Com o distribuidor de alta tensão desactivado, e um digitalizador com a configuração *0xAB* (o dispositivo *TTCrx* dos digitalizadores apresenta o contador de eventos e o contador de *bunch* activos) para o contador de eventos e o contador de *bunch (DigNoise)*.

2. Com o distribuidor de alta tensão activo (mas sem a entrada de alta tensão activa), e um digitalizador com a configuração *0xA8* (ambos o contador de eventos e o contador de *bunch* desactivados) para o contador de eventos e o contador de *bunch (DigNoiseHV)*.

2.3.3.5 Testes *Integ* e *IntegHV*

O objectivo destes testes é verificar a linearidade e o nível de ruído de ADC-I e do circuito integrador de carga das placas *3in1*.

1. O valor RMS do ruído é calculado e analisado para uma centena de medições.
2. A entrada do circuito integrador de carga é ligada a um conversor Digital-Analógico (DAC) das placas de digitalização. A saída do circuito integrador de carga é digitalizada pelo ADC-I. É verificada a linearidade entre a carga digital injectada (DAC) e a resposta digital do ADC-I.

Este teste é realizado duas vezes:

1. Com o distribuidor de alta tensão desactivado.
2. Com o distribuidor de alta tensão activado.

2.3.3.6 Teste *CommHV*

O objectivo deste teste é verificar a comunicação com a electrónica de distribuição de alta tensão:

1. A comunicação com a placa *HV micro* (distribuidor de alta tensão para os PMTs), através de *CanBus,* é estabelecida. Caso haja sucesso na comunicação, o registo de estado global desta placa é lido e verificado.
2. A versão do *software* da placa *HV micro* é verificada.
3. Os valores de baixa tensão e temperaturas monitorizadas pela placa *HV micro* são verificados.
4. Os Números de Série de duas placas *HVopto* e da placa *HV micro* são comparados com os Números de Série esperados para o respectivo *Super-drawer* (obtidos a partir da base de dados).

2.3.3.7 Testes *Opto e NominalHV*

Este teste tem como objectivo verificar a funcionalidade da electrónica de distribuição de alta tensão.
Para o teste *Opto*:

1. Os quatro interruptores HV (par/ímpar interno/externo) são desligados. Depois são ligados e verificados um por um.

2. A alta tensão HV é estabelecida com 700 V em todos os canais. Depois, em cada canal, a tensão medida é lida e comparada com o valor esperado.

3. A alta tensão é estabelecida com 600 V em todos os canais. Mais uma vez, a tensão medida é comparada com o valor esperado.

Para o teste *NominHV*:

1. As altas tensões nominais são restauradas em todos os canais a partir da memória EEPROM da placa *HV micro*.

2. Para cada canal, a tensão medida é comparada com o valor esperado e o valor da tensão (lido de EEPROM) é comparado com o da alta tensão nominal esperado para o respectivo canal, lido da base de dados dos *Super-drawers*.

2.3.3.8 Teste *DigShapeLED*

O objectivo deste teste é verificar a resposta dos *Super-drawer* a impulsos luminosos similares aos impulsos produzidos pelas partículas que atravessam o *TileCal* durante as experiências do LHC. É utilizada neste teste a placa *LED Driver* do MobiDICK. As saídas da placa *LED Driver* são conectadas a duas caixas contendo um LED azul á frente de uma fibra óptica. Os dados digitalizados assim como os impulsos reconstruídos a partir das amostras digitais, são lidos e analisados.

2.4 Conclusão

Neste capítulo foi feita uma descrição da estrutura e do princípio de funcionamento do calorímetro hadrónico, o *TileCal*. Foi também feita a apresentação da electrónica de frontaria do *TileCal* e do testador, o MobiDICK, que é utilizado para testar as funcionalidades da electrónica de frontaria do *TileCal*.

A electrónica de frontaria do *TileCal* encontra-se organizada em estruturas compactas, normalmente designadas de *Drawers*. Um conjunto de 2 *Drawers* forma um *Super-Drawer*. Este sistema electrónico tem por função, realizar a aquisição de sinais do *TileCal* e disponibilizar à electrónica de retaguarda, dados para a leitura.

O testador MobiDICK é um sistema móvel utilizado para o teste e a verificação da funcionalidade e integridade da electrónica de frontaria do *TileCal*. Este testador é constituído

por um componente de *hardware* e por um de *software*, que operam juntos para realizar testes à electrónica de frontaria e comunica com o utilizador através de uma interface *Ethernet*, com recurso ao protocolo TCP/IP. O *hardware* do MobiDICK 4 é implementado utilizando sistemas reconfiguráveis, ao contrário do MobiDICK 3, cujo *hardware* é implementado utilizando placas VME.

Capítulo 3 Implementação e Teste de Interfaces *Ethernet*

Neste capítulo é apresentada a descrição da implementação e teste de duas interfaces *Ethernet* num sistema reconfigurável (numa FPGA – a Virtex 6 da *Xilinx*), controlada por um microprocessador embebido da *Xilinx*, o *MicroBlaze*. A implementação consistiu em duas fases, ambas realizadas utilizando a ferramenta EDK. A primeira fase consistiu no desenvolvimento da componente de *hardware* do sistema de teste, utilizando a plataforma XPS, ao passo que, a segunda fase consistiu na implementação de aplicações de *software* com a ferramenta SDK desenvolvidas em C, para serem executadas pelo microprocessador embebido. A ferramenta SDK inclui algumas bibliotecas, *drivers* e também algumas aplicações de interação com o sistema já finalizadas.

As duas interfaces *Ethernet* implementadas e testadas são disponibilizadas pela *Xilinx* [15, 16]: A interface *Tri-Mode Ethernet Media Access Control* (TMAC) e a interface *Ethernet Lite Media Access Control* (ELM). Estas duas interfaces realizam as funções dos subníveis de *Controlo de Acesso ao Meio* e *Controlo de Ligação Lógica* do modelo de comunicação *Ethernet* (Apêndice A). Ambas estão disponíveis na plataforma de desenvolvimento ISE da *Xilinx*, sendo descritas e especificadas através de linguagens de descrição de *hardware,* VHDL e *Verilog.* Neste trabalho optou-se pela linguagem de descrição de *hardware* VHDL (utilizada também noutros módulos do testador). Na primeira parte deste capítulo proceder-se-á à definição dos termos utilizados e à descrição dos módulos implementados.

3.1 Implementação da Componente de *Hardware*

O diagrama de blocos do sistema implementado na FPGA é constituído pelos seguintes blocos principais: o Microprocessador *(MicroBlaze)*, o Controlador de Interrupções *(XPS INTC)*, o bloco Temporizador/Contador *(XPS Timer/Counter)*, uma Interface *Ethernet* de teste e uma interface UART[3] *(XPS UART LITE)* para comunicação série, representados na figura 3.1.

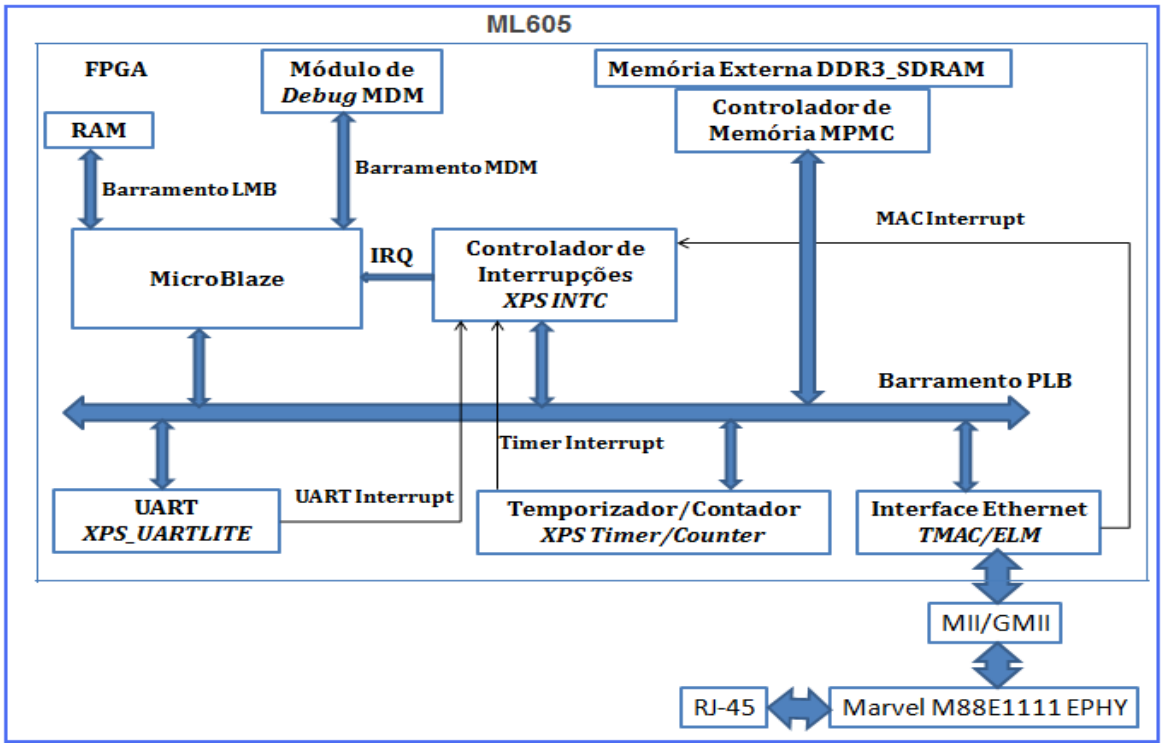


Figura 3.1: *Diagrama de blocos do sistema implementado.*

A placa ML605 (utilizada para a implementação) utiliza um dispositivo *Marvel M88E1111 EPHY* como camada física *(Transceiver)* para as comunicações *Ethernet*. A conexão entre a camada física e um cabo de rede para a conexão a outros dispositivos de comunicação é implementada através de um conector *HFJ11-1G01E RJ-45*. Esta implementação suporta velocidades de comunicação de 10, 100 e 1000 Mbps. Para esta aplicação utilizou-se a interface MII/GMII, disponível na placa ML605, para conectar a interface *Ethernet* (Nível MAC) à camada física. A interface MII/GMII já se encontrava disponível na placa ML605, sendo necessário configurar a placa com as configurações indicadas na tabela 3.1 [17].

Interface	**Configuração *Jumper* ML605**		
	J66	J67	J68
MII/GMII	Sobre os pinos 1-2	Sobre os pinos 1-2	Sem *Jumper*

Tabela 3.1: *Configuração da Placa ML605 para suportar a interface MII/GMII.*

[3] UART- Sigla de *Universal Asynchronous Receiver Transmitter*

Os módulos utilizados para a implementação são disponibilizados na plataforma EDK através núcleos IP *(Intellectual Property)* completamente funcionais. Na tabela 3.2 encontram-se todos os módulos de *hardware* utilizados para a implementação e teste das interfaces TMAC e ELM, assim com as respectivas versões.

Módulo de *Hardware*	**Núcleo IP**	**Versão**
Microprocessador: MicroBlaze	microblaze	8.00.b
Barramento: Processor Local Bus	plb_46	1.05.a
Barramento: Local Memory Bus	lmb_v10	1.00.a
Memória Local: BRAM	Bram_block	1.00.a
Controlador Memória Local	Lmb_bram_if_cntlr	2.10.b
UART	xps_uartlite	1.01.a
Interface Ethernet	xps_ethernetlite	4.00.a
	xps_ll_tmac	2.02.a
Contador (Timer)	xps_timer	1.02.a
Memória Externa+Controlador	mpmc	6.02.a
Módulo de Debug	mdm	2.00.a
Controlador de Interrupções	xps_intc	2.01.a
Relógio	Clock_generator	4.01.a
Sistema de *Reset*	Proc_sys_reset	3.00.a

Tabela 3.2: *Núcleos IP utilizados na implementação.*

3.1.1 Microprocessador Embebido *MicroBlaze*

Um microprocessador é um dispositivo programável que aceita dados binários vindos de um dispositivo de entrada, processa os dados de acordo com as instruções armazenadas na memória (o programa) e fornece um resultado nas linhas de saída do sistema. Por outras palavras, o microprocessador executa o programa na memória e transfere dados de, e para, fora através de portas de entrada/saída.

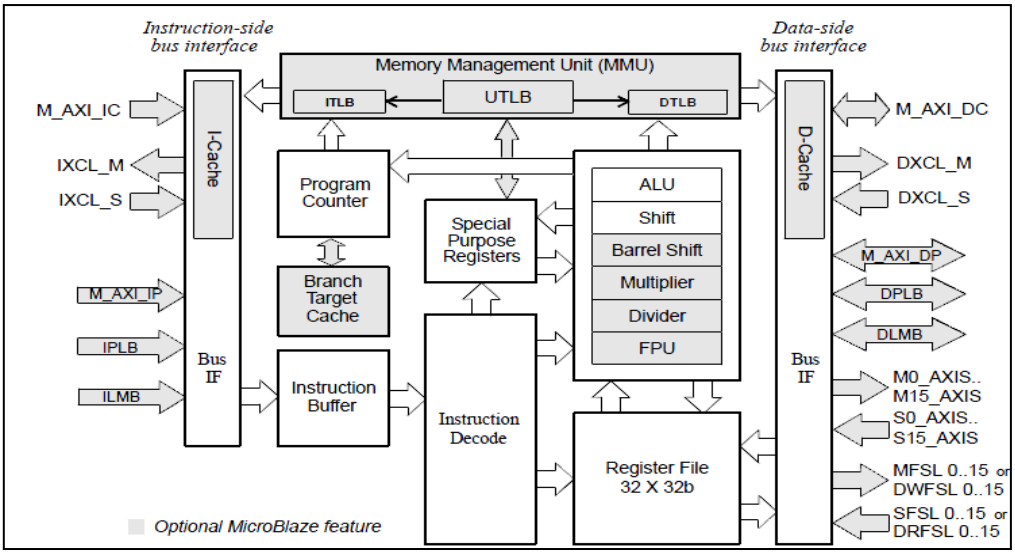


Figura 3.2: *Arquitectura do microprocessador embebido MicroBlaze* [18].

O diagrama da arquitectura do microprocessador *MicroBlaze* encontra-se representado na figura 3.2. O microprocessador embebido *MicroBlaze* é do tipo *Reduced Instruction Set Computer (RISC)*, sendo optimizado para a implementação em FPGAs da *Xilinx* [18]. Este tipo de microprocessador é implementado por forma a apresentar uma certa simplicidade, sendo simultaneamente bastante eficiente e rápido. As implementações de microprocessadores do tipo RISC são realizadas por forma a executarem um conjunto de instruções básicas, necessárias e suficientes, de modo a que permita maximizar a velocidade efectiva de execução de instruções, através da redução do número de ciclos de relógio por instrução [19]. O *MicroBlaze* utiliza os formatos *Big-endian* e *Little-endian* para representar dados dependendo da configuração definida e suporta os seguintes tipos de dados: *word* (32 *bits*), *half word* (16 *bits*) e *byte* (8 *bits*) [18]. Apresenta uma arquitectura do tipo *Harvard* [18], isto é, apresenta separação entre barramentos de dados e de instruções. O *Microblaze* apresenta interfaces a diversos tipos de barramento, neste trabalho utilizou-se o barramento *Processor Local Bus v4.6* da *Xilinx.* O *MicroBlaze* foi configurado, utilizando a ferramenta EDK, com uma frequência de 100 MHz e com memória *cache* desactivada.

3.1.2 Barramento Local do Processador

Um barramento é um sistema que permite a interligação entre o microprocessador e os vários periféricos de um sistema embebido ou de outros sistemas de computação, tais como computadores de uso geral. Neste trabalho foi utilizado o barramento *Processor Local Bus v4.6* da *Xilinx* de 128 *bits* e que opera a 100 MHz (o mesmo barramento utilizado no testador), obtendo-se uma infraestrutura que permitiu conectar um conjunto de periféricos *Master (microprocessador)* e um conjunto de periféricos *Slave (UART, Interface Ethernet, etc)*. Este barramento consiste de uma unidade de controlo, de um temporizador *Watchdog,* de uma unidade de leitura e escrita de dados separadas e de uma unidade opcional chamada de *Device Control Register (DCR)* que permite o acesso aos registos de estados de erros [20].

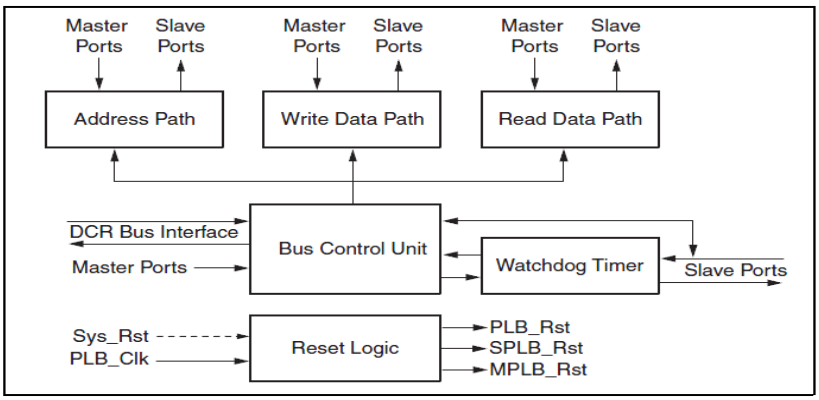


Figura 3.3: *Diagrama de blocos do barramento PLB* [20].

A seguir são descritas os blocos constituintes do barramento PLB [20]:

Endereçamento

Este bloco contem o mecanismo necessário para selecionar o endereço principal *(Master)* que é atribuído a dispositivos *Slave*, na saída de endereços do barramento.

Escrita de Dados

Esta unidade contém a lógica necessária para a escrita de dados dos dispositivos *Master* e *Slave*.

Leitura de Dados

Esta unidade contém a lógica necessária para a leitura de dados dos dispositivos *master* e *slave*.

Unidade de Controlo do Barramento

Esta unidade, tal como o nome indica, controla o barramento. Consiste de uma unidade de arbitragem que controla o fluxo de endereços e dados através do barramento e da unidade DCR.

Temporizador *Watchdog*

Este temporizador tem a função gerar um sinal de *Time out (PLB_MTimeout)* quando não há resposta de dispositivos *Slave* a uma solicitação do *Master.* O tempo de *Time out* é definido como sendo 16 ciclos de relógio.

Lógica de *Reset*

Esta unidade permite a sincronização do sinal de *reset* externo (*Sys_Rst)* com os sinais de *reset* do barramento PLB *(PLB_Rst, SPLB_Rst* e *MPLB_Rst)*. O barramento PLB não apresenta nenhum circuito que permita que um sinal de *reset* seja gerado quando o sistema for ligado, sendo necessário um sistema de *reset* externo ao barramento. Isto implica que uma implementação do barramento PLB seja acompanhada de um sistema de *reset* externo, sendo normalmente utilizado o módulo *Proc_sys_reset*, um módulo de *reset* disponível na plataforma EDK, para garantir que o sinal *reset* é gerado no mínimo em 16 ciclos de relógio, depois do sistema ter sido ligado.

3.1.3 Barramento Local da Memória

O barramento local da memória permite a conexão entre o microprocessador *MicroBlaze* e um bloco de memória local (RAM[4]). Este barramento permite o acesso rápido à memória. Permite conexões separadas para dados e instruções com blocos de memória de duas entradas. As principais características deste barramento são [21]: eficiente, único *master* (não requer arbitragem), apresenta barramentos separados de leitura e escrita e utiliza poucos recursos na FPGA.

3.1.4 Controlador de Interrupções

Uma interrupção é um sinal enviado ao microprocessador que o faz parar a execução de uma dada tarefa e executar outra tarefa especificada através da interrupção. Ou seja, o microprocessador para de executar o programa principal e executa outro código.

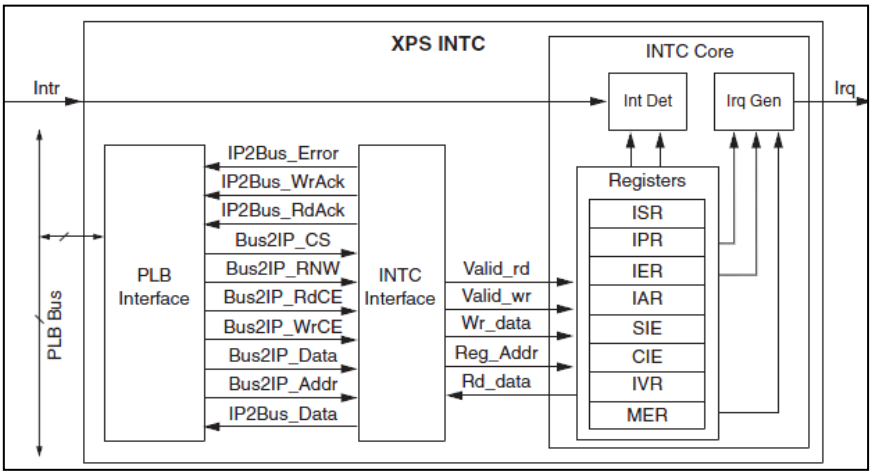


Figura 3.4: *Arquitectura do controlador de interrupções XPS INTC* [22].

Num sistema embebido onde há vários periféricos, cada periférico pode eventualmente gerar interrupções, o que faz com que seja necessário um controlador de interrupções. Para este trabalho utilizou-se o controlador de interrupções da *Xilinx LogiCORE IP XPS Interrupt Controller (XPS INTC).* O módulo *XPS INTC* aceita várias entradas interrupções vindas de periféricos, e gera um único sinal de interrupção de saída para o microprocessador. O controlador *XPS INTC* é utilizado para expandir o número de interrupções disponíveis para o microprocessador, e opcionalmente, fornece um esquema de codificação de prioridades de interrupções. A saída de *XPS INTC* deve ser ligada à entrada de interrupção do microprocessador, e cada entrada de interrupção para o *XPS INTC* conecta aos dispositivos capazes de gerar interrupções [22]. A

[4] RAM – Sigla de *Random Access Memory*

figura 3.4 ilustra o diagrama de blocos do controlador de interrupções *XPS INTC* da *Xilinx*, que é constituído pelos seguintes blocos [22]:

Interface do Barramento Local do Processador (PLB Interface)

Este bloco fornece uma interface para o barramento PLB permitindo a transferência de dados entre o processador e o controlador de interrupções *XPS INTC*.

Interface INTC

É um bloco lógico que permite conectar o bloco *INTC Core* ao bloco de Interface ao barramento PLB.

Núcleo Controlador de Interrupções *(INTC Core)*

Este bloco apresenta 3 subcomponentes:

- **Detecção de Interrupção *(Int Det)*:** permite a detecção de um sinal de interrupção activo enviado por um periférico.
- **Registos:** Contêm todos os estados e registos de controlo. Este controlador de interrupções contém registos acessíveis ao programador permitindo que interrupções sejam activadas, consultadas ou desactivadas por controlo a nível de *software*.
- **Geração de Pedidos de Interrupções *(Irq Gen)*:** Gera o sinal de interrupção de saída (IRQ) para o microprocessador.

3.1.5 Temporizador/Contador

Este bloco funciona como temporizador e contador, sendo necessário à implementação do protocolo TCP/IP. Quando se implementa um protocolo, é necessário programar, agendar ou contabilizar eventos. O bloco Temporizador/Contador é utilizado para programar eventos, determinar intervalos de tempo e contabilizar eventos. Neste trabalho foi utilizado o módulo *LogiCORE IP XPS Timer/Counter.* Na figura 3.5 encontra-se o diagrama de blocos deste módulo, onde há dois blocos principais [23]:

- **Módulo de Interface com o PLB (*PLB Interface Module*):** representa uma interface que permite conectar o *XPS Timer/Counter* ao barramento PLB.
- **Temporizador/Contador *(Timer/Counter)*:** representa o bloco principal de XPS *Timer/Counter* que realiza a programação e contagens de eventos.

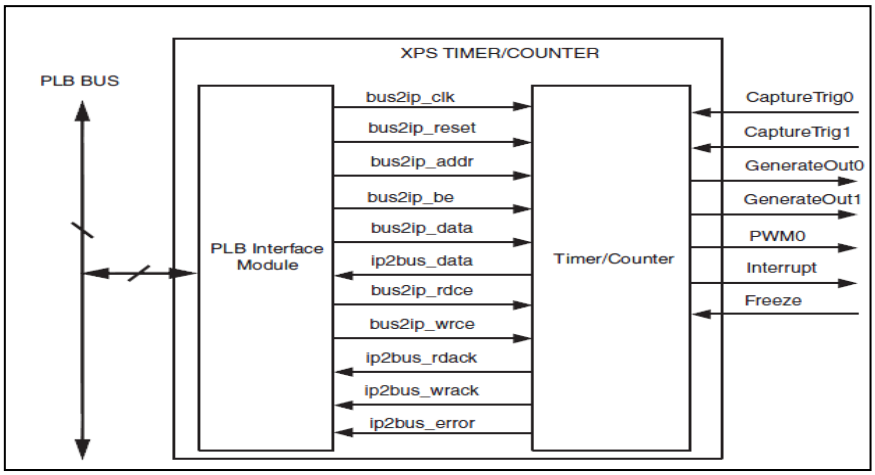


Figura 3.5: *Diagrama de blocos do módulo XPS Timer/Counter* [23].

3.1.6 Interface Ethernet

As interfaces *Ethernet* implementadas na FPGA foram duas: a interface *Tri-Mode Media Access Controller* e a interface *Ethernet Lite Media Access Controller.* Estas interfaces *Ethernet* são disponibilizadas pela *Xilinx* através da plataforma EDK.

3.1.6.1 Interface *Tri-Mode Media Access Controller (TMAC)*

A interface *TMAC* suporta três valores de velocidades de ligação, 10/100/1000 Mbps. Suporta também os dois modos *half-duplex* e *full-duplex,* sendo configurável para permitir a selecção quer do modo de funcionamento quer da velocidade de ligação. A implementação desta interface foi realizada de acordo com a especificação da norma IEEE 802.3-2008 [15]. A seguir é apresentada a descrição da respectiva arquitectura e dos principais blocos que a constituem.

A figura 3.6 representa o diagrama de blocos funcional da arquitectura da interface *TMAC*. Os principais blocos que constituem esta interface *Ethernet* são [15]:

Interface Cliente *(Client Interface)*

Este bloco é implementado para permitir a máxima flexibilidade para a ligação de uma interface ao microprocessador (para esta aplicação ao barramento PLB).

Bloco de Transmissão *(Transmit Engine)*

Este bloco implementa o mecanismo de transmissão de dados para a interface MII/GMII *(Gigabit/Media Independent Interface)*. Os dados enviados através da Interface Cliente são processados neste bloco e convertidos no formato GMII/MII. Este bloco é também responsável pela adição dos campos *Preamble* e *Frame Check Sequence* no pacote *Ethernet*, e realiza também

o processo de *Padding* se isso for necessário para garantir que o pacote de dados irá ter a dimensão mínima requerida. Fornece também estatísticas do processo e transmite pacotes *Ethernet* de pausa, quando estes forem gerados pelo módulo do controlo de fluxo de dados.

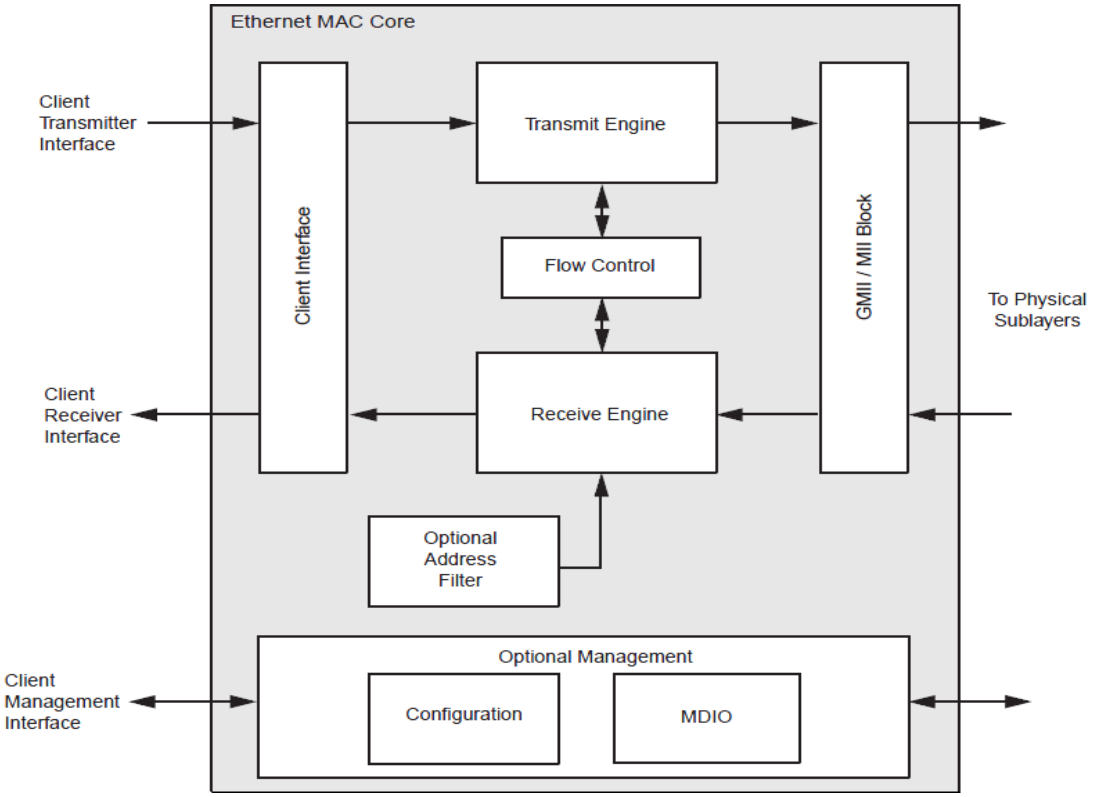


***Figura 3.6**: Diagrama de blocos da interface TMAC.*

Bloco de Recepção *(Receive Engine)*

Este bloco recebe dados da interface GMII/MII (enviados da camada física) e realiza o processo de verificação, se está de acordo ou não com a especificação IEEE 802.3. O campo *Padd* é removido e os dados são enviados para a Interface Cliente. Este bloco colecta estatísticas relativas aos pacotes recebidos.

Controlo do Fluxo *(Flux Control)*

Este bloco foi implementado de acordo com a norma IEEE 802.3-2008, cláusula 31. A interface TMAC pode ser configurada para enviar pacotes de pausa com um valor programável de pausa, e actuar quando os recebe. Estes dois processos podem ser configurados assimetricamente.

Bloco *MDIO Interface*

É uma interface opcional utilizada para a configuração de dispositivos da camada física. É uma interface definida de acordo com a norma IEEE 802.3 na cláusula 22.

Bloco GMII/MII

Este bloco implementa a interface de interacção com a camada física, recebe dados do bloco de transmissão e converte-os no formato MII se a velocidade de conexão for inferior a 1 Gbps e converte-os para o formato GMII se a velocidade de conexão for igual ou superior a 1 Gbps.

Interface de Monitorização *(Management Interface)*

Esta interface é opcional, para um processador independente com endereços de dados e sinais de controlo padrão. Este bloco serve para a configuração e monitorização das interfaces TMAC e MDIO.

Filtro de Endereços *(Address Filter)*

É um bloco opcional de filtragem de endereços MAC. Se estiver activo, o dispositivo elimina os pacotes *Ethernet* que não contenham um grupo de endereços selecionados pelo cliente.

3.1.6.2 Interface *Ethernet Lite Media Access Controller (ELM)*

A interface *Ethernet Lite Media Access Controller* oferece a funcionalidade mínima necessária para implementar uma interface *Ethernet* com a utilização de recursos mínimos [16]. Esta interface é implementada de acordo com a norma IEEE 802.3 *Media Independent Interface* e permite uma capacidade de transmissão de dados de 10 Mbps e 100 Mbps. A figura 3.7 representa o diagrama de blocos desta interface, sendo seguidamente descrita a funcionalidade de cada um dos respectivos blocos [16]:

Transmissão

Este bloco implementa a lógica de transmissão de pacotes, sendo constituído por um módulo Gerador de CRC, um Multiplexador de Transmissão *(MUX)*, *TX FIFO*, por um bloco de Controlo de Transmissão *(Transmit Control)* e por uma Interface de Transmissão *(TX Interface)*. O módulo CRC calcula o valor de CRC para os pacotes *Ethernet* a serem transmitidos. O módulo *MUX* ordena o pacote, envia os campos *Preambule*, *Start-of-Frame Delimiter*, campo *Data*, *Padd* e *CRC*, para o módulo *TX FIFO* na ordem correcta. O pacote *Ethernet* é enviado à camada física (PHY), após esta operação o bloco *Transmit control* gera um sinal de interrupção *(IP2INTC_Irpt)* e actualiza os registos de controlo de transmissão.

Recepção

Este módulo implementa a lógica de recepção de pacotes de dados, sendo constituído por um bloco Multiplexador *(Loop Back Mux)*, *RX FIFO*, um bloco verificador de CRC *(CRC Checker)*, um módulo de Controlo de Recepção de pacotes *(Receive Control)* e por uma interface de recepção *(RX Interface)*. Os dados recebidos da camada física passam pelo *Loop Back Mux* e são armazenados no bloco *RX FIFO*. Se a opção *loop back* for selecionada os dados do bloco *TX FIFO* são passados para *RX FIFO*. O módulo verificador de CRC calcula o valor de CRC de pacotes *Ethernet* recebidos. Se o valor correcto de CRC for obtido para um pacote, o bloco *Receive Control*, gera um sinal de interrupção *(IP2INTC_Irpt)* de um pacote recebido sem erros.

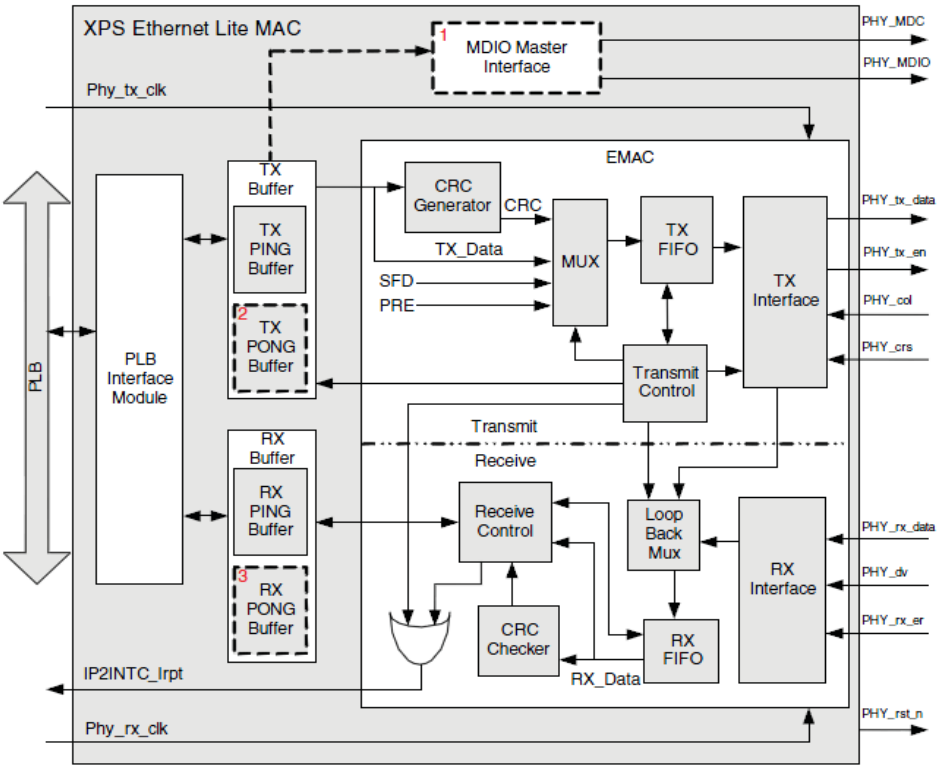


Figura 3.7: *Arquitectura da interface ELM.*

Módulo da Interface *PLB*

Este bloco implementa a interface entre a ELM e o barramento PLB. Realiza o protocolo necessário entre o barramento PLB e ELM, permitindo o acesso pelo microprocessador.

Bloco Tampão *TX (TX Buffer)*

Este módulo é um bloco de memória de 2 kBytes, com dois portos, para armazenar os dados de um pacote *Ethernet* completo e também os registos de controlo da interface de transmissão *(Tx Interface)*. Inclui também um bloco de memória opcional *(Tx Pong Buffer)*.

Bloco Tampão *RX (RX Buffer)*

Corresponde a um bloco de memória de 2 kBytes, com dois portos, para armazenar dados contidos num pacote *Ethernet* completo e registos de controlo da interface de recepção *(Rx interface).* Possui também uma memória opcional de 2 kbyte com dois portos *(RX Pong Buffer).*

Interface MDIO Master

Este módulo é opcional e fornece o acesso aos registos da camada física para a gestão da respectiva camada ou do dispositivo físico.

3.1.7 Bloco UART

O bloco UART permite uma transferência assíncrona de dados em série. Este bloco foi utilizado na implementação para depuração de aplicações de *software* executadas pelo microprocessador embebido. Os resultados de execução de programas e depuração são transferidos (em série, através deste bloco) da placa ML605 (através da porta UART) para o computador (com um terminal apropriado para a visualização desses resultados). O computador recebe os dados através de uma porta *USB*. Foi utilizado para a implementação o módulo *XPS UART Lite (v1.01a)* disponível na plataforma EDK. A figura 3.8 representa o diagrama de blocos desta interface, cujo principais blocos são [24]:

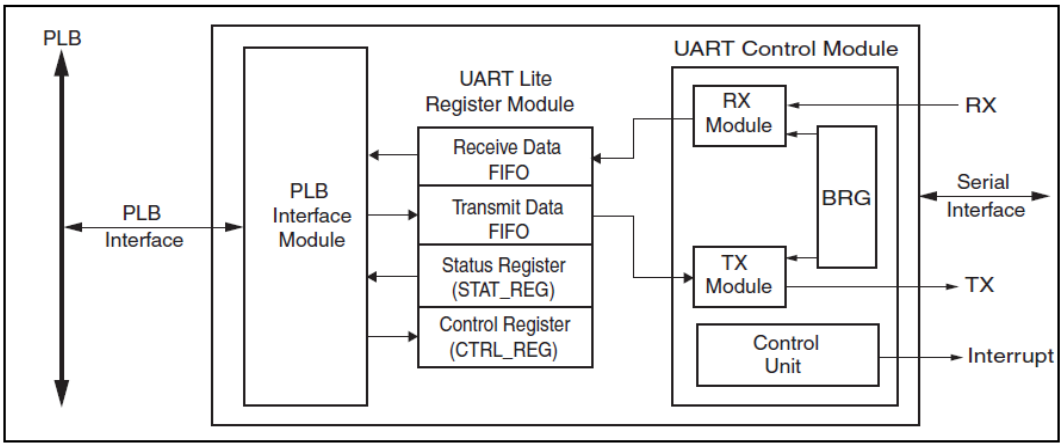


Figura 3.8: *Diagrama de blocos de XPS UART Lite (v1.01a)* [24].

Módulo de Controlo UART

Este bloco inclui um módulo *RX* para a recepção de dados, um módulo *TX* para transmissão de dados, um gerador de *Baud rate* parametrizável e uma unidade de controlo *(Control Unit)* responsável pela geração de sinais de interrupção. Incorpora a máquina de estados para inicialização, e a lógica de controlo do bit *start* e *stop*.

Módulo da Interface PLB

Este módulo fornece uma interface de acesso ao barramento PLB, implementando o protocolo necessário de acesso ao barramento.

Módulo de Registo *UART Lite*

Este bloco tem acesso ao barramento através do módulo da interface PLB. Apresenta um registo de estado de 8 *bits*, um registo de controlo de 8 *bits* e um par de blocos FIFO de 8 *bits* para guardar dados de Transmissão/Recepção. Todos esses registos são acedidos a partir do barramento PLB e através do módulo da interface PLB.

3.1.8 Controlador de Memória *MPMC* e Memória Externa

Foi utilizada neste trabalho uma memória externa na qual o acesso é realizado através do controlador de memória *Multi-Port Memory Controller* (MPMC) da *Xilinx*. O controlador de memória MPMC é totalmente parametrizável e suporta os seguintes tipos de memória externa: SDRAM[5], DDR[6], DDR2, DDR3 e LPDDR[7] [25]. Este controlador dispõe de 8 portas de acesso à memória permitindo conexões a 8 periféricos. Apresenta interfaces para o *MicroBlaze* utilizando o barramento local do processador (PLB) ou o barramento *Xilinx Cache Link*. A memória externa aqui utilizada foi do tipo DDR3, presente na placa ML605. A sua utilização permite colmatar a limitação da memória local (bloco RAM) ser de apenas 64 kBytes, assegurando um melhor desempenho do sistema.

3.2 Aplicações de *Software*

As aplicações de *software* utilizadas para o teste das interfaces *Ethernet* com recurso ao protocolo TCP/IP, dividem-se em dois grupos: o grupo de aplicações desenvolvidas para serem executadas pelo microprocessador embebido (servidor/cliente) na placa ML605; e as aplicações instaladas e executadas no computador de teste. As aplicações de *software* utilizadas para o teste e validação das duas interfaces *Ethernet* (TMAC e ELM) são baseadas nos exemplos indicados na referência [26]. Essas aplicações são programas implementados em C que podem ser compiladas na plataforma SDK.

---

[5] SDRAM - Sigla de *Synchronous Dynamic Random Access Memory*
[6] DDR – Sigla de *Double Data Rate*
[7] LPDDR - Sigla de *Low Power Double Data Rate Memory*

Testes		Placa ML605	Computador
Desempenho	Transmissão	Cliente *lwIP*	Servidor *Iperf*
	Recepção	Servidor *lwIP*	Cliente *Iperf*
Fiabilidade		Servidor de eco	*PING*
Controlo da Placa via *Ethernet*		Servidor *Web*	*Browser*
Depuração		-	*TeraTerm*

Tabela 3.3: *Aplicações de software executados na placa ML605 e no computador de teste.*

A tabela 3.3 sumariza o conjunto de aplicações utilizadas para o teste de comunicação utilizando as interfaces Ethernet implementadas.

Aplicações Executadas na Placa ML06

Do conjunto daquelas referidas em [26], para a placa ML605 utilizou-se a aplicação servidor de eco *(echo server)*, que devolve qualquer *bit* enviado à interface, permitindo o teste de fiabilidade de comunicação, uma aplicação de comunicação com o computador utilizando uma aplicação de nome *Iperf*, previamente instalada no computador para a determinação de taxas de transmissão e recepção de dados e um servidor *Web* que permitiu controlar e monitorizar componentes de *hardware* da placa ML605. Para a implementação dessas aplicações de comunicação, utilizou-se como base o conjunto de protocolos TCP/IP, que são disponibilizados na plataforma XPS, através de uma implementação chamada de *lightweight Internet Protocol* (lwIP).

O protocolo TCP/IP representa um conjunto de protocolos que fornecem funções ao nível de rede e de transporte do modelo de referência OSI (Apêndice A). O nível de rede é definido com o protocolo IP, que realiza a função de roteamento, isto é, a cada dispositivo de uma rede é atribuído um endereço único, habitualmente designado de endereço IP. Numa rede densa, com vários percursos disponíveis para a transmissão de dados, o protocolo IP tem a funcionalidade de selecionar e estabelecer a melhor rota possível para transmissão de dados, permite também a gestão e manutenção das rotas de comunicação. O nível de transporte é definido com o protocolo TCP *(Transmission Control Protocol)* ou com o protocolo UDP *(User Datagram Protocol)*. O protocolo TCP permite um mecanismo de entrega fiável dos dados, ou seja, é função deste protocolo garantir que os dados cheguem garantidamente ao destino sem erros. O protocolo UDP implementa também um mecanismo de transporte, mas não fiável de dados, pois não apresenta o mecanismo de garantia de entrega de dados.

O protocolo TCP/IP é normalmente implementado como um serviço de um sistema operativo. Para sistemas embebidos é normalmente utilizada a referida implementação chamada de *lightweight Internet Protocol* (lwIP), a qual representa uma implementação sem a necessidade de utilização de um sistema operativo, embora possa ser utilizada num sistema

operativo. A plataforma EDK já inclui a implementação do lwIP e uma forma fácil de adicioná-lo ao sistema, permitindo construir aplicações de comunicação baseadas em TCP/IP. A implementação de lwIP fornece suporte aos seguintes protocolos de comunicação [27]:

- *Internet Protocol* (IP)
- *Internet Control Message Protocol* (ICMP)
- *User Datagram Protocol* (UDP)
- *Transmission Control Protocol* (TCP)
- *Address Resolution Protocol* (ARP)
- *Dynamic Host Configuration Protocol* (DHCP)

A implementação de lwIP fornece duas formas de programar as aplicações de comunicação baseadas no protocolo TCP/IP, isto é, duas formas de aceder aos serviços fornecidos pelo protocolo TCP/IP, o modo *RAW* e o modo *Socket* [27]:

- ***RAW API*:** é baseado no método de *callback*. As aplicações obtêm acesso directo às funções do conjunto TCP/IP e vice-versa. O modo *Raw* fornece um excelente desempenho devido à interacção directa com o protocolo. Permite a implementação sem a necessidade de instalar um sistema operativo.

- ***Socket API*:** fornece um estilo *BSD socket*. Fornece um modelo de execução em modo de bloqueio, num paradigma *open-read-write-close*. A biblioteca em modo *socket* requer a utilização do sistema operativo *Xilkernel* sendo a programação baseada no modelo concorrente de *threads*. Face ao modo *Raw*, apresenta um desempenho fraco.

Aplicações Executadas no Computador

O *Iperf* é uma aplicação para o teste de desempenho de redes de comunicação desenvolvida utilizando a linguagem de programação C++ pela *National Laboratory for Applied Network Research* (NLANR) dos Estados Unidos da América [28]. Esta aplicação permite determinar o desempenho de uma rede onde é utilizado o protocolo TCP/IP. A *Xilinx* recomenda a sua utilização para o teste de comunicação utilizando as suas interfaces *Ethernet* com recurso a lwIP. Esta aplicação foi utilizada para a comunicação com a placa ML605 para a determinação do desempenho da comunicação utilizando as interfaces *Ethernet* em teste com recurso ao protocolo TCP/IP. A aplicação *Iperf* permite a comunicação no modelo Cliente/Servidor e pode funcionar como cliente e também como servidor. Esta aplicação foi bastante útil para este trabalho, pois é de fácil utilização, eficiente, confiável e de acesso livre. A sua utilização é realizada através do terminal de comandos disponibilizado pelo sistema operativo.

A fiabilidade de comunicação das interfaces *Ethernet* e a respectiva acessibilidade foram testadas utilizando o teste normalmente designado por PING *(Packet INternet Groper).* O teste PING é utilizado para testar a acessibilidade a um dispositivo numa rede que utiliza o protocolo IP como protocolo do nível de rede do modelo de comunicação. O sistema operativo instalado *(Windows 7)* no computador de teste já inclui a funcionalidade para o teste PING.

A aplicação *TeraTerm* é um terminal de comunicação capaz de ligar-se a outros sistemas através de TCP/IP, ou através de portas de comunicação série. Neste trabalho foram utilizados para a depuração de programas executados pelo microprocessador embebido. Esta aplicação conecta à placa ML605, permitindo uma comunicação série com o sistema embebido implementado através da interface *xps_uartlite,* servindo-se da porta UART da placa ML605. O *TeraTerm* fornece uma interface gráfica permitindo a sua configuração e conexão ao sistema embebido de forma relativamente fácil. São também aplicações livres.

3.3 Conclusão

Neste capítulo foi feita a apresentação do sistema implementado na FPGA, para o teste de duas interfaces *Ethernet* disponibilizadas pela *Xilinx,* como núcleo IP na ferramenta de desenvolvimento EDK.

Foram implementadas e testadas duas interfaces *Ethernet*: *Tri-Mode Media Access Controller* (TMAC) e *Ethernet Lite Media Access Controller* (ELM). A interface TMAC suporta três valores de velocidades de ligação, 10/100/1000 Mbps e suporta os dois modos de funcionamento *half-duplex* e *full-duplex,* sendo configurável para permitir a selecção quer do modo de funcionamento quer da velocidade de ligação. A implementação da interface TMAC foi realizada de acordo com a especificação da norma IEEE 802.3-2008. A interface ELM oferece a funcionalidade mínima necessária para implementar uma interface *Ethernet* com a utilização de recursos mínimos. Esta interface é implementada de acordo com a norma IEEE 802.3 *Media Independent Interface* e permite uma capacidade de transmissão de dados de 10 Mbps e 100 Mbps.

A par da componente de *hardware* implementada utilizando a ferramenta EDK, foram utilizadas algumas aplicações de *software* implementadas em C e disponibilizadas pela *Xilinx*. Essas aplicações permitiram o teste de comunicação entra a placa ML605 e um computador.

Capítulo 4 Implementação do Módulo CRC

Este capítulo é dedicado à descrição da implementação, no testador MobiDICK 4, de um algoritmo de verificação da integridade dos dados enviados pela electrónica de frontaria do calorímetro hadrónico *TileCal*. Durante a realização dos testes utilizando o MobiDICK 4, a Placa de Interface da electrónica de frontaria envia dados pelas ligações ópticas ao MobiDiCK 4. O testador dispõe de um emulador *G-Link*, capaz de receber esses dados mediante o envio do comando L1A. Esses dados encontram-se organizados como um pacote de dados específico do sistema. O algoritmo implementado será capaz de detectar e contabilizar os erros nos pacotes de dados enviados pela Placa de Interface da electrónica de frontaria. Nesta já se encontra implementado esse algoritmo (o chamado *front-end*), a electrónica de retaguarda também dispõe de um algoritmo análogo, o *back-end.*

Um algoritmo de verificação de integridade de dados encontrava-se já implementado na electrónica de retaguarda (ROD), em FPGAs da *Altera*. Este algoritmo apenas verifica a integridade dos dados numa situação real de aquisição de dados durante as experiências do LHC. O testador MobiDICK 4 deverá também dispor de um algoritmo análogo quando efectua testes, que para além da deteção de erros, permita também a sua contabilização. O sistema ROD utiliza FPGAs da *Altera*, mas no MobiDICK 4 é utilizada uma placa ML507 equipada com uma FPGA da *Xilinx*, pelo que foi necessário efectuar algumas adaptações e bastantes simulações com as ferramentas da *Xilinx* antes da implementação do algoritmo no MobiDICK 4. O algoritmo de verificação de dados é conhecido por *Cyclic Redundany Check* (CRC). Este algoritmo é bastante utilizado em sistemas de comunicação como, por exemplo, na *Ethernet* com TCP/IP, e em ligações USB, etc.

Neste capítulo é realizada uma introdução ao algoritmo que efectua o CRC, é feita a descrição do formato do pacote de dados enviados da electrónica de frontaria e é feita também uma descrição da implementação do módulo CRC no MobiDICK 4.

4.1 Verificação de Integridade de Dados - *Cyclic Redundancy Check*

A verificação de integridade de dados digitais baseada no algoritmo CRC é bastante utilizada em sistemas de comunicação. Este algoritmo permite que o receptor de uma mensagem, transmitida através de um canal que apresenta ruído, determine se a mensagem foi corrompida ou não. Para isso, o transmissor da mensagem calcula um parâmetro chamado valor de CRC ou *Checksum,* que é função da mensagem a transmitir e anexa-o à respectiva mensagem. O receptor utiliza a mesma função e operações para o cálculo do valor de CRC ou *Checksum* da mensagem recebida e compara-o com o *Checksum* anexado à mensagem, verificando se são ou não iguais. Se os dois valores de *Checksum* forem compatíveis considera-se que não houve erros na transmissão, caso contrário, considera-se que houve erros e a mensagem é normalmente descartada. A figura 4.1 ilustra o processo de verificação da integridade de dados baseado no algoritmo CRC, num sistema de comunicação digital.

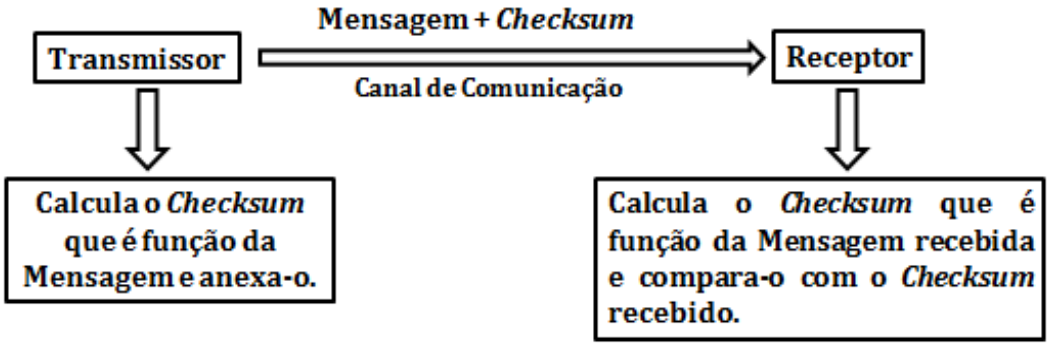


Figura 4.1: *Princípio de verificação da integridade de dados num sistema de comunicação utilizando o CRC.*

Num sistema de comunicação digital, a mensagem corresponde a um conjunto de *bits*, consequentemente, no cálculo do *Checksum* ou CRC a mensagem é tratada como uma enorme sequência binária. O cálculo de *Checksum* é simples e baseia-se em operações binárias de módulo 2. Normalmente essas sequências binárias são representadas como polinómios de coeficientes binários (0 ou 1), sendo efectuadas operações em aritmética de módulo 2, utilizando os respectivos coeficientes. O cálculo de CRC consiste em 3 passos fundamentais:

1. Escolha de um polinómio (coeficientes binários) específico de acordo com a aplicação pretendida; este polinómio é normalmente designado de Polinómio Gerador.

2. Multiplica-se (multiplicação binária de módulo 2) o polinómio correspondente à mensagem pelo termo de maior grau do Polinómio Gerador.

3. Divide-se o resultado da operação realizada anteriormente em (2) pelo Polinómio Gerador. Da divisão binária de módulo 2 resulta um quociente e um resto. O resto é considerado o valor de CRC ou *Checksum* da mensagem, o quociente é descartado.

O algoritmo descrito pode ser melhor compreendido através de um exemplo. Suponhamos que numa comunicação digital entre dois dispositivos, a mensagem a ser transmitida é a sequência binária 1001. Esta sequência binária é representada através do seguinte polinómio de coeficientes binários:

$$M(x) = 1 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0 = x^3 + 1$$

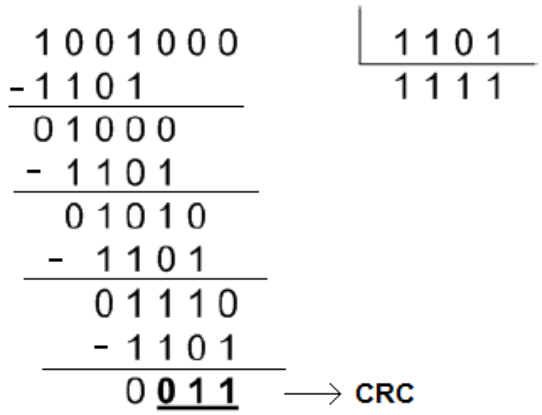
Supondo que o polinómio gerador escolhido é representado pela sequência binária 1101, dado pelo seguinte polinómio:

$$P(x) = x^3 + x^2 + 1$$

O valor de CRC será dado por,

$$CRC = Resto\left[\frac{M(x) \cdot x^3}{P(x)}\right]$$

A operação de multiplicação (módulo 2) consiste simplesmente num aumento do grau de $M(x)$ em 3 unidades, de acordo com o grau do Polinómio Gerador. O algoritmo da divisão (módulo 2) para este caso é apresentado a seguir:



Este valor de CRC é anexado à mensagem original, formando a sequência binária [1001 011]. Na verificação da integridade de dados num sistema de comunicação, este algoritmo é implementado tanto no Transmissor como no Receptor, utilizando o mesmo Polinómio Gerador e as mesmas operações, para que seja possível a comparação entre valores de *Checksum* calculados por ambos. Espera-se que o Receptor da mensagem obtenha o mesmo valor de CRC,

se a mensagem não for corrompida pelo ruído existente no canal de comunicação. Na divisão, a principal operação é a subtracção, no qual corresponde à operação binária OU-EXCLUSIVO (XOR). Existem diferentes Polinómios Geradores que podem ser utilizados de acordo com a aplicação pretendida. Na tabela 4.1 é apresentado alguns dos algoritmos CRC mais utilizados, bem como os respectivos Polinómios Geradores.

Designação	**Polinómio Gerador**	**Notação Hexadecimal**
CRC-12	$x^{12} + x^{12} + x^3 + x^2 + x + 1$	80F
CRC-16	$x^{16} + x^{15} + x^2 + 1$	8005
CRC-CCITT	$x^{16} + x^{12} + x^5 + 1$	1021
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	04C11DB7

Tabela 4.1: *Algoritmos CRC frequentemente utilizados.*

A dimensão em *bits* do valor do *Checksum* é sempre igual ao grau do Polinómio Gerador. A electrónica de frontaria utiliza os polinómios CRC-16 e CRC-CCITT na verificação da integridade de dados.

4.2 Processo de Verificação da Integridade de Dados do *TileCal*

Conforme foi descrito no capítulo 2, o calorímetro hadrónico *TileCal* é constituído por 3 blocos cilíndricos. São este o bloco central *(Long Barrel)* e dois blocos laterais *(Extended Barrels)*, cada um constituído por 64 módulos. Cada módulo apresenta 48 posições para instalar fotomultiplicadores (PMTs), mas apenas 45 são utilizados no bloco central e apenas 32 nos blocos laterais. Os sinais luminosos produzidos pelos cintiladores são convertidos em sinais eléctricos pelos fotomultiplicadores. Esses sinais são transmitidos para as placas de digitalização através das referidas placas *3in1*, onde esses sinais analógicos são digitalizados. Há 8 placas de digitalização por cada módulo, e cada placa contém 2 dispositivos chamados DMU *(Data Management Unit)* (16 DMUs por módulo). A figura 4.2 ilustra esta arquitectura na electrónica de frontaria.

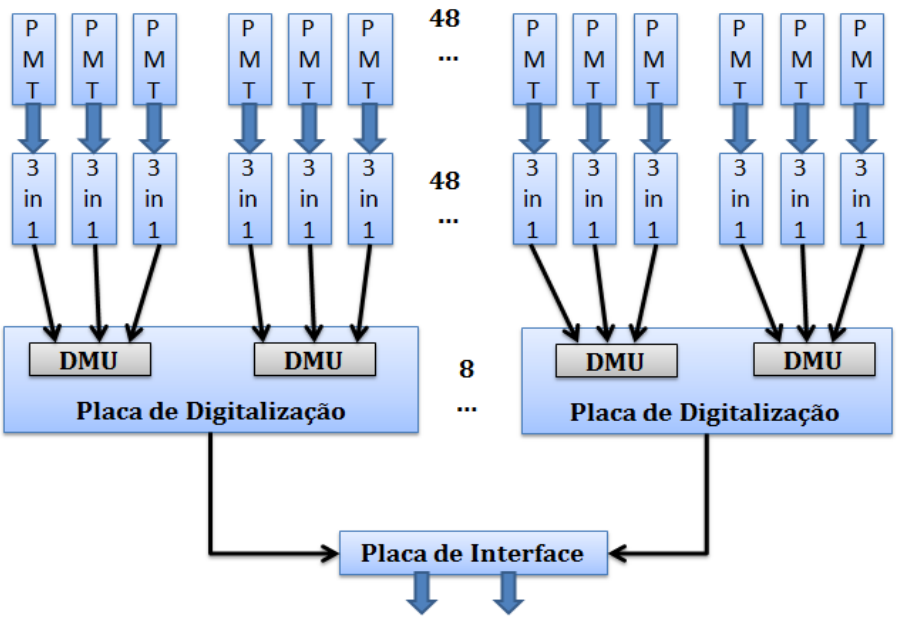


Figura 4.2: *Diagrama da electrónica de frontaria do TileCal.*

Cada DMU é responsável pela formatação dos sinais digitalizados em pacotes de dados, provenientes de cada 3 canais. Cada dispositivo DMU, depois de construir seu o pacote de dados envia-o á Placa de Interface, que por sua vez constrói um pacote global contendo os dados de todos os dispositivos DMUs. Cada DMU calcula os valores de CRC (*bits* pares e ímpares) e anexa-os ao seu respectivo pacote e envia este para a Placa de Interface, que por sua vez calcula novamente esses valores de CRC para cada pacote DMU recebido e compara-os com os valores anexados ao pacote. Se houver uma transmissão correcta de dados entre todos os dispositivos DMUs e a Placa de Interface, ela constrói um pacote global contendo os dados de todos os DMUs,

e calcula um novo valor de CRC, denominado de *CRC Global*, que é anexado ao pacote global. A Placa de Interface envia o pacote global para o ROD, ou em situações de teste envia-o para o MobiDICK 4. No sistema ROD calcula-se novamente os valores de CRC de cada pacote DMU e o valor de *CRC Global*, comparando-os com os valores recebidos.

No sistema MobiDICK 4 será implementado o mesmo algoritmo de cálculo de CRC e verificação de integridade do sistema ROD. A figura 4.3 ilustra e resume o processo da verificação da integridade de dados transmitidos entre a electrónica de frontaria e a de retaguarda.

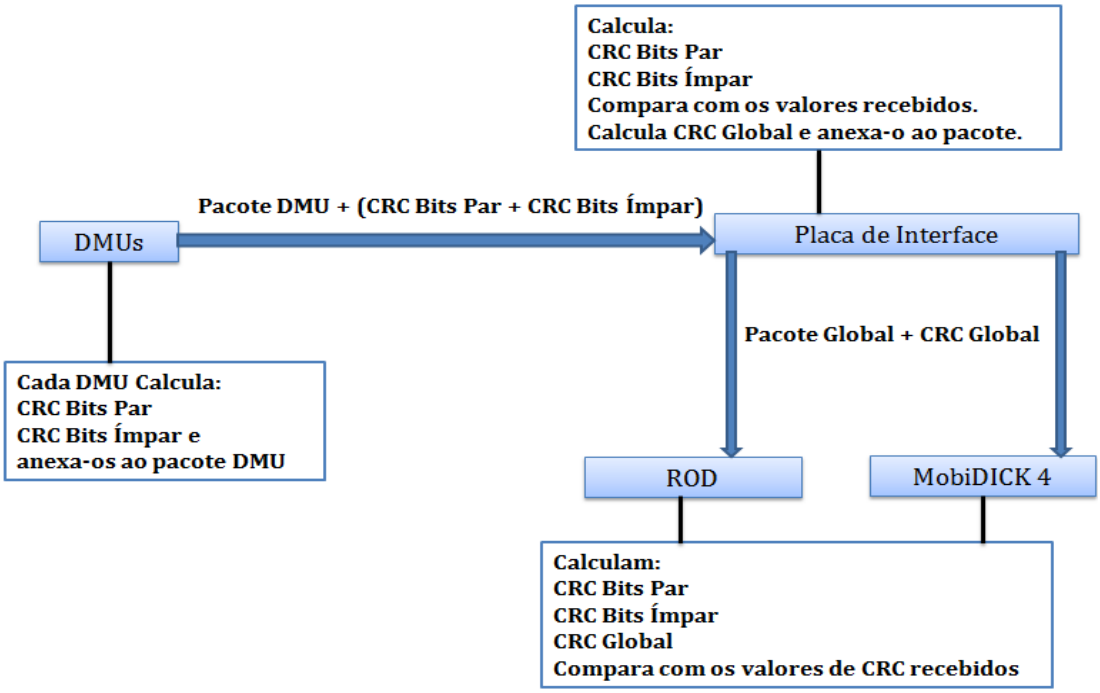
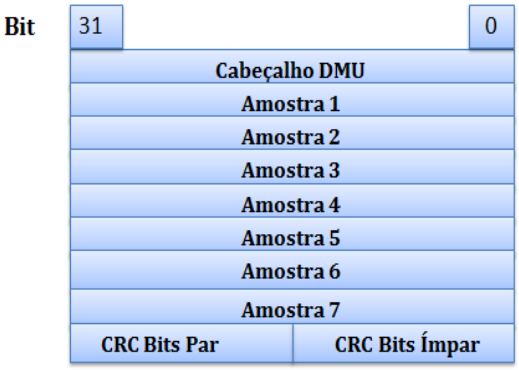


Figura 4.3: *Esquema de verificação da integridade de dados entre a frontaria e a retaguarda implementado no TileCal.*

4.2.1 Pacote de Dados DMU

O pacote de dados de cada DMU (figura 4.4) é constituído por palavras digitais de 32 *bits*. A primeira palavra representa o cabeçalho comum a todos os dispositivos DMU. As outras palavras digitais consistem nas amostras obtidas da digitalização dos sinais analógicos provenientes dos PMTs. A última palavra digital contém duas informações: os primeiros 16 *bits* mais significativos contêm o resultado do cálculo de valor de CRC em função dos *bits* pares do cabeçalho e das amostras digitais; os outros 16 *bits* menos significativos contêm o valor de CRC calculado em função dos *bits* ímpares do cabeçalho e das amostras digitais. Cada DMU calcula os CRCs e anexa-os ao pacote antes de enviar este para a Placa de Interface.



(a) Ganho '0'

(b) Ganho '1'

Figura 4.4: *Pacotes de dados construído por cada DMU.*

Para o ganho '1', a última palavra digital não contém nenhum tipo de informação útil sendo denominada de *Dummy word*: não é utilizada para a verificação da integridade de dados através do algoritmo CRC, serve apenas para a identificação do fim do pacote.

4.2.2 Pacote de Dados da Placa de Interface

Cada DMU transmite o seu pacote de dados à Placa de Interface. Esta constrói um pacote global (figura 4.5) que inclui os 16 pacotes dos DMUs e duas outras palavras digitais de 32 *bits*, uma delas chamada de *DMU_Chip_mask* e a outra correspondente ao valor global de CRC, calculado em função dos 16 pacotes de dados dos 16 DMUs.

1	**Cabeçalho Global**
2	Cabeçalho DMU 1
3	DMU 1: Amostra 1
4	DMU 1: Amostra 2
5	DMU 1: Amostra 3
6	DMU 1: Amostra 4
7	DMU 1: Amostra 5
8	DMU 1: Amostra 6
9	DMU 1: Amostra 7
10	CRC DMU 1
11	Cabeçalho DMU2
12	DMU 2: Amostra 1
13	DMU 2: Amostra 2
14	DMU 2: Amostra 3
15	DMU 2: Amostra 4
16	DMU 2: Amostra 5
17	DMU 2: Amostra 6
18	DMU 1: Amostra 7
19	CRC DMU 2
...	...
137	Cabeçalho DMU 16
138	DMU 16: Amostra 1
139	DMU 16: Amostra 2
140	DMU 16: Amostra 3
141	DMU 16: Amostra 4
142	DMU 16: Amostra 5
143	DMU 16: Amostra 6
144	DMU 16: Amostra 7
145	CRC DMU 16
146	*Dmu_chip_mask_word*
147	CRC Global
148	*Trailer*

(a) Ganho '0'

1	**Cabeçalho Global**
2	Cabeçalho DMU 1 (Baixo Ganho)
3	DMU 1: Amostra 1 (Baixo Ganho)
4	DMU 1: Amostra 2 (Baixo Ganho)
5	DMU 1: Amostra 3 (Baixo Ganho)
6	DMU 1: Amostra 4 (Baixo Ganho)
7	DMU 1: Amostra 5 (Baixo Ganho)
8	DMU 1: Amostra 6 (Baixo Ganho)
9	DMU 1: Amostra 7 (Baixo Ganho)
10	Cabeçalho DMU 1 (Alto Ganho)
11	DMU 1: Amostra 1 (Alto Ganho)
12	DMU 1: Amostra 2 (Alto Ganho)
13	DMU 1: Amostra 3 (Alto Ganho)
14	DMU 1: Amostra 4 (Alto Ganho)
15	DMU 1: Amostra 5 (Alto Ganho)
16	DMU 1: Amostra 6 (Alto Ganho)
17	DMU 1: Amostra 7 (Alto Ganho)
18	DMU 1: *Dummy*
...	...
265	Cabeçalho DMU 16 (Alto Ganho)
266	DMU 16: Amostra 1 (Alto Ganho)
267	DMU 16: Amostra 2 (Alto Ganho)
268	DMU 16: Amostra 3 (Alto Ganho)
269	DMU 16: Amostra 4 (Alto Ganho)
270	DMU 16: Amostra 5 (Alto Ganho)
271	DMU 16: Amostra 6 (Alto Ganho)
272	DMU 16: Amostra 7 (Alto Ganho)
273	DMU 16: *Dummy*
274	*Dmu_chip_mask_word*
275	CRC Global
276	*Trailer*

(b) Ganho '1'

Figura 4.5: *Formato do pacote de dados construído pela Placa de Interface.*

A palavra digital *Cabeçalho Global* é utilizada, no nível ROD para detectar o início do pacote e a última palavra digital *(Trailer)* é utilizada para detectar o seu fim. A Placa de Interface calcula o valor global de CRC de 16 *bits* para o pacote global e anexa-o nos últimos 16 *bits* menos significativos do campo *CRC Global* de 32 *bits*. O pacote é enviado para o ROD preenchendo os 16 *bits* mais significativos do campo *CRC Global* com zeros. O sistema ROD, ao receber o pacote calcula novamente o valor de CRC do pacote global e substitui os 16 *bits* mais significativos nulos pelo valor obtido, sendo realizada a comparação com o valor enviado nos 16 *bits* menos significativos. Se os dois valores forem compatíveis significa que houve uma transmissão correcta de dados entre a electrónica de frontaria e a electrónica de retaguarda.

A Placa de Interface calcula novamente o valor de CRC, tanto para *bits* pares e ímpares de cada DMU, e compara com o valor enviado por cada DMU. Se os resultados de CRC calculados forem compatíveis com os valores enviados por cada DMU, a palavra digital *DMU_Chip_mask* é sinalizada com o valor lógico '1'. Isto significa que, se houver uma transmissão correcta entre os 16 DMUs e a Placa de Interface, a palavra digital *DMU_Chip_mask* terá todos os *bits* iguais a '1' correspondente em hexadecimal a *0xFFFFFFFF* para o bloco central e *0x0FFF0FFF* para os blocos laterais, onde são implementados apenas 12 DMUs por módulo.

4.3 Implementação

Quando o LHC está em funcionamento regular, a Placa de Interface comunica com o sistema ROD, mas durante a realização de testes ela irá comunicar com o testador MobiDICK através de ligações ópticas (utilizando o emulador *G-Link*). Isto significa que o algoritmo de verificação da integridade de dados implementado no sistema ROD deverá ser o mesmo implementado no MobiDICK 4. O ROD utiliza FPGAs da *Altera*, onde está implementado o algoritmo CRC em VHDL. Um dos objectivos do trabalho por nós realizado foi implementar o mesmo algoritmo CRC, na placa de controlo do MobiDICK 4, a placa ML507 equipada com uma FPGA Virtex-5 da *Xilinx*. Esta implementação consistiu numa adaptação do algoritmo em VHDL implementado na FPGA da *Altera* para uma FPGA Virtex-5 da *Xilinx*. A primeira fase do trabalho consistiu na simulação do algoritmo existente utilizando as ferramentas da *Xilinx*, com o objectivo de verificar o seu correcto funcionamento. Após as primeiras simulações, concluiu-se que seriam necessárias algumas modificações e adaptações ao código. Foram também adicionados em VHDL contadores para contagens dos erros associados aos pacotes dos DMU e ao pacote Global. A fase seguinte consistiu na integração do algoritmo no sistema embebido do MobiDICK 4 que contou com a colaboração da equipa IFIC-UV[8], e finalmente realizou-se o teste

[8] IFIC-UV-Sigla de *Instituto de Física Corposcular – Universidade de Valencia*

do módulo CRC numa situação real de aquisição de pacotes de dados enviados pela Placa de Interface da electrónica de frontaria.

4.3.1 Estrutura do Módulo CRC

O módulo CRC, implementado em VHDL, encontra-se estruturado em componentes (figura 4.6). Apresenta um componente principal *(crc_check_link)* com os sinais de entrada e saída do módulo CRC. Internamente, o componente principal é constituído por dois subcomponentes, um para o cálculo do *CRC Global* e detecção dos erros associados a ele *(crc_full_link)* e o outro para o cálculo do valor de CRC de cada DMU e a detecção dos erros associados a estes *(Dmu_crc_check).*

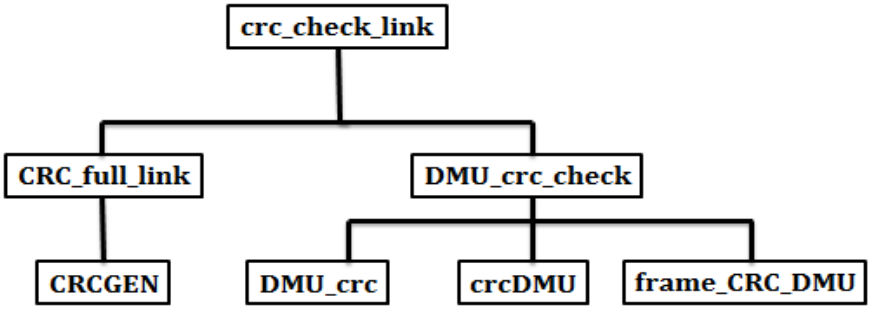


Figura 4.6: *Estrutura do módulo CRC implementado na plataforma ISE da Xilinx.*

4.3.1.1 Módulo Principal: *crc_check_link*

Este módulo representa o nível de topo (figura 4.7) e consiste do módulo principal. Os sinais de entrados estão indicados à esquerda, e os sinais de saída à direita.

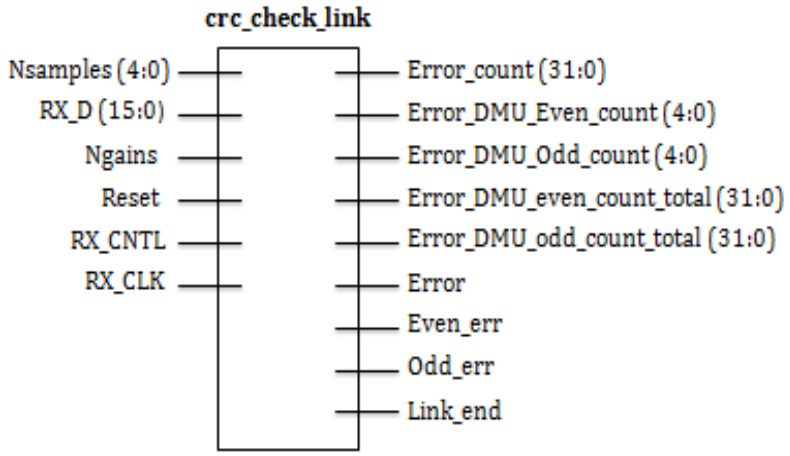


Figura 4.7: *Módulo principal do projecto do CRC.*

Internamente o módulo principal é constituído pelos dois módulos internos: *crc_full_link* e *Dmu_crc_check,* assim como se pode verificar na figura 4.8.

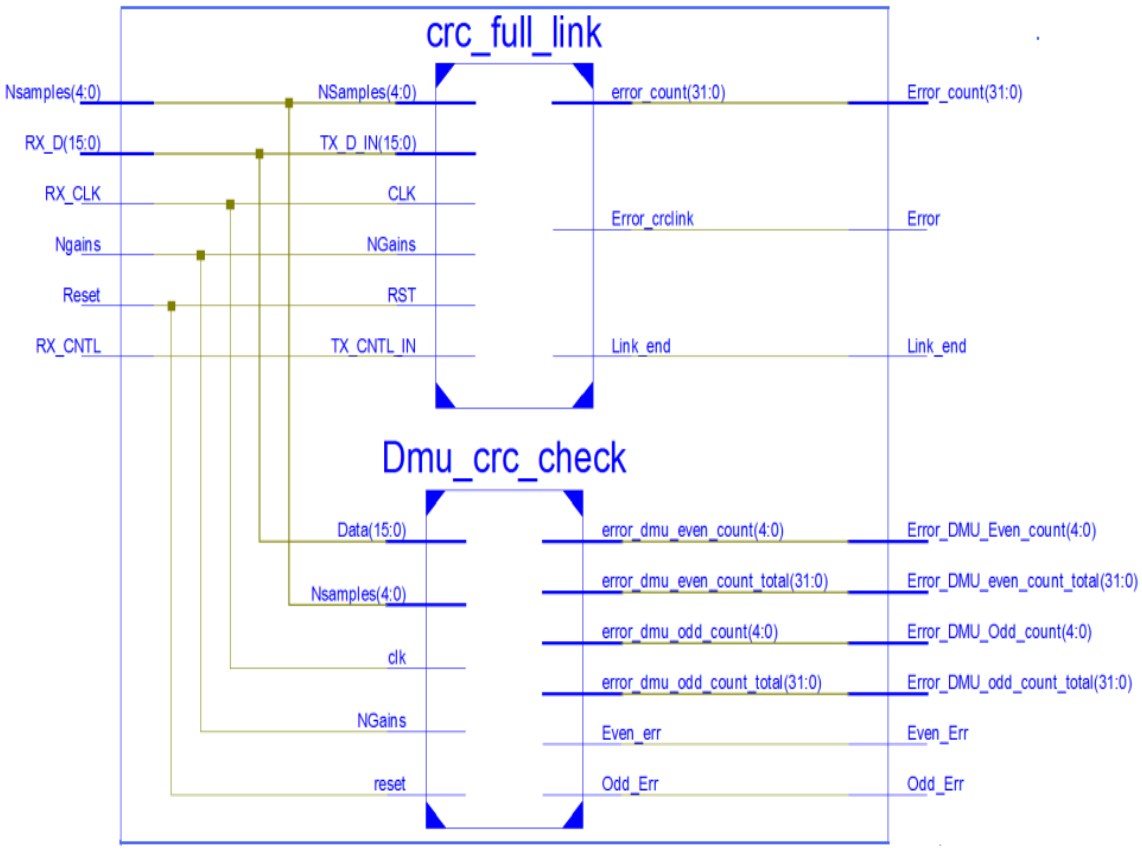


Figura 4.8: *Estrutura interna do módulo CRC.*

O sinal *Nsamples* indica o número de amostras por DMU, que é também determinado pela escolha do ganho através do sinal *Ngains*. O sinal *RX_D* indica o canal para a entrada de palavras digitais de 16 *bits* do pacote de dados. O sinal de *Reset* permite reinicializar o sistema. O sinal *RX_CLK* é o sinal de relógio, cuja frequência é escolhida de acordo com a frequência de envio de pacotes dos *Super-drawers*. A frequência de envio de pacotes da electrónica de frontaria é, aproximadamente 100 kHz. Uma vez que cada palavra digital de 16 *bits* do pacote de dados é processada em cada ciclo de relógio pelo módulo CRC, o relógio *RX_CLK* deverá ter frequência superior. Nos primeiros testes utilizou-se uma frequência de 40 MHz. Os dois módulos internos são síncronos com o módulo principal, partilhando o mesmo relógio. O sinal *RX_CNTL* representa um sinal de controlo inicializado a zero, que permite activar o cálculo de CRC quando o cabeçalho for detectado, pois um dos *bits* do cabeçalho funciona como activação do sinal *RX_CNTL*. A descrição resumida dos sinais de entrada encontra-se na tabela 4.2.

Sinais de Entradas	Largura *(bits)*	Descrição
Nsamples	5	Número de amostras por DMU
RX_D	16	Palavras digitais do pacote de dados
Ngains	1	Ganho
Reset	1	Sinal de *Reset*
RX_CLK	1	Sinal de Relógio
RX_CNTL	1	Sinal de Controlo

Tabela 4.2: *Descrição dos sinais de entrada do módulo CRC.*

Nesta aplicação foram implementados 5 contadores binários, para permitir a contagem dos erros associados aos valores de CRC, cujos sinais de saída encontram-se descritos na tabela 4.3.

Sinais de Saídas	Largura *(bits)*	Descrição
Error_count	32	Sinal de saída de um contador binário dos erros associados ao CRC Global de todos os pacotes.
Error_DMU_Even_count	5	Sinal de saída de um contador binário dos erros associados ao CRC DMU (*bits* par) de cada pacote.
Error_DMU_Even_count_total	32	Sinal de saída de um contador binário dos erros associados ao CRC DMU (*bits* par) de todos os pacotes.
Error_DMU_Odd_count	5	Sinal de saída de um contador binário dos erros associados ao CRC DMU (*bits* ímpar) de cada pacote.
Error_DMU_Odd_count_total	32	Sinal de saída de um contador binário dos erros associados ao CRC DMU (*bits* ímpar) de todos os pacotes
Error	1	Indica erros associados ao CRC Global
Even_err	1	Indica erros associados ao CRC DMU (*bits* par).
Odd_err	1	Indica erros associados ao CRC DMU (*bits* ímpar).
Link_end	1	Indica o fim *(Trailer)* de um pacote.

Tabela 4.3: *Descrição dos sinais de saída do módulo CRC.*

O sinal *Error* serve para a indicar a detecção de erros associados ao *CRC Global*, e adquire o valor lógico '1' se detectar um erro num pacote de dados e valor '0', no caso contrário. Os sinais *Even_err* e *Odd_err* permitem indicar a detecção de erros associados aos valores de CRC de DMUs, para *bits* pares e ímpares, respectivamente. Adquirem o valor lógico '1' em caso de detecção de erros e '0' em caso contrário. O sinal *Link_end*, indica a última palavra digital de um pacote de dados, toma o valor lógico '1' no fim de um pacote e '0', caso contrário.

4.3.1.2 Módulo: *crc_full_link*

Este módulo apresenta 3 funcionalidades:

1. Cálculo do valor do *CRC Global* do pacote de dados recebido. Apresenta um subcomponente chamado *CRCGEN*, o qual é responsável pelo cálculo do *CRC Global*.

2. Detecção de erros associado ao valor do *CRC Global*, através da comparação entre o valor calculado com o valor anexado ao pacote de dados, apresenta o sinal de saída *Error* para sinalizar erros no *CRC Global*.

3. Realização de contagem dos erros associados ao valor do *CRC Global*. Apresenta um sinal de saída de 32 *bits*, *Error_count,* o que disponibiliza o valor do contador interno.

Neste módulo é implementada uma máquina de estado que detecta a palavra digital recebida.

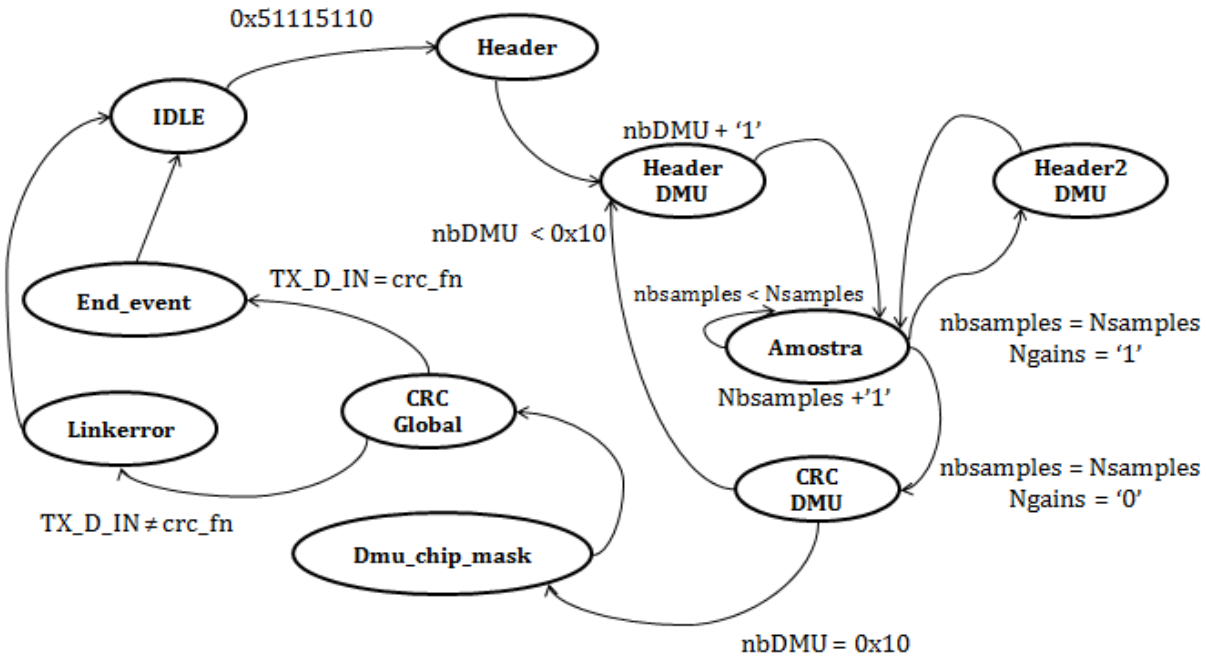


Figura 4.9: *Máquina de estados para a detecção da palavra digital recebida da electrónica de frontaria.*

A figura 4.9 mostra a máquina de estados, implementada no módulo CRC do sistema ROD (módulo *crc_full_link*) com o objectivo de detectar a palavra digital enviada pela electrónica de frontaria do *TileCal* e detectar a ocorrência de erro no *CRC Global*. Esta máquina de estados inicia-se com o estado *IDLE*, onde permanece até receber uma palavra digital dos *Super-drawer*. Assim que receber uma palavra digital com o cabeçalho (*0x51115110* em hexadecimal), transita para o estado *Header*, que por sua vez transita para o estado *Header DMU* (correspondente ao

cabeçalho do pacote de DMUs). Sempre que é detectado o cabeçalho de um pacote DMU, é incrementado o sinal *nbDMU* que corresponde a um contador de DMUs, sendo que o número máximo de contagens esperado é 16 (*0x10* em hexadecimal) que é o número de DMUs por *Super-drawer* do *TileCal*. Quando as palavras digitais correspondentes aos sinais digitalizados começam a ser recebidas, transita-se ao estado *Amostras*, de onde se sai apenas quando o sinal *nbsamples* for igual ao número de amostras predefinido *(Nsamples)*. O sinal *nbsamples* corresponde a um contador de amostras. Quando a electrónica de frontaria termina de enviar as amostras de um determinado DMU, isto é, para $nbsamples = Nsamples$, há duas situações a considerar: ganho '0' ou '1'.

1. Se o ganho for igual a '0', significa que a próxima palavra digital a ser recebida é o valor de CRC do repectivo DMU, ocorrendo uma transição de estado para *CRC DMU*. No estado *CRC DMU* (correspondente à ultima palavra digital de um pacote determinado DMU) é necessário verificar se este CRC corresponde ao do último pacote DMU ou não. Para tal, é utilizado um contador *nbDMU*, se for $nbDMU = 16$ *(0x10)* significa que o valor de CRC em questão pertence ao último pacote DMU, transitando imediatamente para o estado *Dmu_chip_mask* (que corresponde a palavra digital recebida imediatamente a seguir ao valor de CRC do último pacote DMU).

2. Se o ganho for igual a '1' a próxima palavra digital recebida corresponde ao segundo cabeçalho de um determinado pacote DMU, ocorrendo uma transição para o estado *Header2 DMU,* de onde se transitará para o estado *Amostras.*

Imediatamente a seguir à palavra digital *Dmu_chip_mask,* é enviado o valor de *CRC Global*, e nesse estado é efectuada a comparação entre o valor de CRC *(crc_fn)* calculado e o valor de CRC enviado *(TX_D_IN):*

1. Se $TX\_D\_IN = crc\_fn$, significa que houve uma transmissão correcta de dados entre a electrónica de frontaria e o testador, e entre os *TileDMUs* e a Placa de Interface, dando-se uma transição para o estado *End_event*, assim assinalando o fim do pacote de dados. Do estado *End_event* volta-se ao estado *IDLE*, para a recepção de mais um pacote de dados, repetindo-se novamente todo o processo.

2. Se $TX\_D\_IN \neq crc\_fn$, significa que não houve uma transmissão correcta de dados, e dá-se uma mudança para o estado *Linkerror*, o que assinala a ocorrência de um erro na transmissão. Nesta situação é activado o sinal *found* com o valor lógico '1'. Quando o contador de erros detectar a activação do sinal *found* incrementa o seu valor de uma unidade. Do estado *Linkerror* transita-se para o estado *IDLE*, ficando o sistema

preparando-se para a recepção de mais um pacote e pronto a repetir novamente todo o processo.

CRCGEN

Este componente utiliza o algoritmo CRC-CCITT16 (tabela 4.1) para o cálculo do valor de *CRC Global*, esse mesmo algoritmo encontra-se implementado na Placa de Interface da electrónica de frontaria do *TileCal*, assim como no sistema ROD e fornece um valor de CRC de 16 *bits*, sendo utilizado um valor inicial dos registos igual a 0xFFFFF. Este módulo foi desenvolvido por *Erik Brandin (CERN, EP-Division)* [29].

Dimensão	16 *bit*
Polinómio	1021 (hexadecimal) - Polinómio Gerador
Valor Inicial	FFFF (Valor inicial do registo)
ReflectIN	*Bits* de entrada não são reflectidos
ReflectOut	Ordem de *bits* do valor de CRC é invertida
XorOut	Nenhuma operação XOR é realizada ao valor de CRC
Check	CRC da *string* ASCII "123456789" é 29B1 (hex)

Tabela 4.4: *Especificações do algoritmo CRC implementado na electrónica de frontaria do TileCal, para o cálculo de CRC Global.*

A tabela 4.4 mostra as especificações mais relevantes do algoritmo CRC implementado na electrónica de frontaria do *TileCal*. A especificação *ReflectIN* permite especificar se a ordem dos *bits* das palavras digitais de entrada é invertida ou não (do bit menos significativo ao bit mais significativo). A especificação *ReflectOut* permite especificar se a ordem dos *bits* do valor de CRC obtido é invertida ou não. A especificação *XorOut* indica se é realizada um *XOR* ao valor CRC antes de o utilizar para a verificação. A especificação *Check* está associada à verificação.

4.3.1.3 Módulo: *Dmu_crc_check*

Este módulo apresenta as 3 principais funcionalidades seguintes:

1. Cálculo dos valores de CRC associados aos *bits* pares e ímpares das palavras digitais do pacote de dados vindo de cada DMU.

2. Detecção dos erros associados aos CRCs dos DMUs, através da sua comparação com os valores de CRC anexados aos pacotes dos DMUs.

3. Realização de contagens dos erros associados aos CRCs de cada DMU por cada pacote recebido, mas também para todos os pacotes, para *bits* pares e ímpares.

crcDMU

É responsável pelo cálculo dos CRCs de cada DMU, para *bits* pares e ímpares. Neste módulo é implementado o processo de verificação da ocorrência ou não de erros associados aos valores de CRC de cada DMU. Contém também 4 contadores, já referidos anteriormente para a contagens de erros, sendo utilizados nesta fase apenas 2 desses contadores. Este módulo é implementado de tal forma que aceita palavras digitais de 32 *bits* na entrada. A electrónica de frontaria envia as palavras digitais de 32 *bits* divididas a meio, constituindo palavras digitais de 16 *bits*. Para que este módulo realize as suas funções correctamente é necessário agregar as palavras de 16 *bits* em palavras digitais de 32 *bits*. Para isso é implementado o módulo *DMU_crc*, descrito de seguida.

Dimensão	16 *bits*
Polinómio	8005 (hexadecimal) - Polinómio Gerador
Valor Inicial	0000 (Valor inicial do registo)
ReflectIN	*Bits* de entrada são reflectidos
ReflectOut	Ordem de *bits* do valor de CRC não é invertida
XorOut	Nenhuma operação *XOR* é realizada ao valor de CRC
Check	CRC da *string* ASCII "123456789" é BB3D (hex)

Tabela 4.5: *Especificações do algoritmo CRC para o cálculo de CRC associados aos DMUs, implementado na electrónica de frontaria do TileCal.*

Na tabela 4.5 encontram-se as especificações mais relevantes do algoritmo implementado na electrónica de frontaria para o cálculo dos CRCs de pacotes de dados vindos dos DMUs e consequentemente implementado no testador MobiDICK 4. O cálculo do CRC para cada DMU (*bits* pares e ímpares) é realizado utilizando o algoritmo CRC16, que utiliza um polinómio de grau 16 e por conseguinte, um valor de CRC de 16 *bits*. O valor inicial dos registos é igual a *0x0000*. Este mesmo algoritmo encontra-se implementado na Placa de Interface da electrónica de frontaria e também no ROD. O cálculo e a verificação de CRC são realizados separadamente em função dos *bits* pares e dos ímpares.

DMU_crc

É um módulo interno a *DMU_crc_check*, e tem a função de organizar as palavras digitais de 16 *bits*, enviadas pela electrónica de frontaria, em palavras digitais de 32 *bits*, necessários ao módulo *crcDMU*, o qual é responsável pelo cálculo dos valores de CRC, para *bits* ímpares e pares.

frame_CRC_DMU

É um módulo onde é implementada uma máquina de estados semelhante à da figura 4.9, que serve para detectar a palavra digital e permitir a verificação dos valores de CRC, tanto para *bits* pares como para os ímpares.

4.3.2 Integração no MobiDICK 4

O módulo CRC foi integrado no emulador *G-Link*, um sistema capaz de receber os dados digitais enviados pela Placa de Interface da electrónica de frontaria através de fibras ópticas. Esta integração contou com a colaboração da equipa da Universidade de Valência, responsável pela implementação do emulador *G-Link*. Foram criados registos no módulo *G-Link* para guardar os resultados dos contadores de erros associados aos valores de CRC do módulo CRC, para que esses registos (com a informação dos contadores) sejam acedidos pelo microprocessador embebido, que é o responsável pelo controlo de todos os módulos do sistema embebido implementado no MobiDICK 4.

4.4 Conclusão

Neste capítulo foi apresentado o módulo para cálculo de CRCs implementado no testador MobiDICK 4. Foi descrito o processo de verificação de integridade de dados num sistema de comunicação utilizando o algoritmo *Cyclic Redundancy Check* (CRC) e particularmente, o processo implementado no *TileCal*. Este algoritmo permite que o receptor de uma mensagem, transmitida através de um canal que apresenta ruído, determine se a mensagem foi corrompida ou não.

No *TileCal* é implementado um esquema de verificação da integridade de dados enviados pela electrónica de frontaria para os sistemas na electrónica de retaguarda. Em modo de teste, a electrónica de frontaria comunica com o testador MobiDICK, e consequentemente o mesmo algoritmo implementado na electrónica de retaguarda foi implementado no MobiDICK 4. Este algoritmo verifica a integridade de dados enviados pela electrónica de frontaria e realiza a contagens de erros.

Capítulo 5 Testes e Resultados

Este capítulo é dedicado à descrição dos testes efectuados e à apresentação dos resultados obtidos no teste das interfaces *Ethernet* e no teste do módulo CRC implementado no testador MobiDICK 4.

Foram duas as interfaces *Ethernet* implementadas no decurso deste trabalho (descritas no capítulo 3), tendo sido realizados diversos testes com o objectivo de escolher qual delas seria implementada no testador. A interface TMAC suporta três velocidades de ligação, 10, 100 e 1000 Mbps, tendo sido realizados testes para cada uma dessas capacidades de ligação. É possível programar a interface TMAC para o mecanismo de autonegociação que permite a determinação da velocidade de conexão automaticamente, bastando apenas configurar o computador com a velocidade de conexão pretendida, de entre os valores possíveis. Em relação à interface ELM, que geralmente apresenta uma capacidade de máxima de 100 Mbps, efectuou-se também o teste para ligações a 10 Mbps. Realizaram-se também a cada um dos modos de programar as aplicações (ou de interacção com os serviços) disponibilizados pela biblioteca lwIP: o modo *Socket* e o modo *RAW.* São também apresentados os resultados do teste de fiabilidade e de acessibilidade das interfaces, com a ferramenta PING. Finalmente, é também apresentado um exemplo de controlo da Placa ML605 através de *Ethernet*, utilizando um servidor *Web* disponibilizado pela *Xilinx*.

Neste capítulo, são também apresentados os resultados obtidos na simulação, teste e validação do módulo CRC, implementado no MobiDICK 4. Para a simulação utilizou-se o simulador ISIM, ao passo que para o teste e validação utilizou-se a ferramenta *Chipscope Analyzer,* juntamente com a aplicação de teste *(Test-stress)* disponibilizada pelo grupo de trabalho e colaboração da Universidade de Valência, implementada em C++ e executada pelo microprocessador embebido.

5.1 Testes das Interfaces Ethernet

Fez-se o teste de comunicação entre o computador de teste (PC) e a placa ML605 através das interfaces *Ethernet* implementadas num sistema embebido que executa o respectivo protocolo. O objectivo destes testes consistiu na determinação das velocidades de transmissão e recepção de dados e na obtenção de resultados relacionados com a acessibilidade às interfaces e com a fiabilidade de comunicação.

A conexão física entre o PC e a placa ML605 fez-se através de um cabo de comunicação *Ethernet* capaz de suportar velocidades de transmissão de 1 Gbps. Para realizar os testes foram utilizadas duas aplicações instaladas no computador: uma aplicação de comunicação com a placa ML605, a aplicação *Iperf* e uma outra aplicação, o *TeraTerm*. A aplicação *TeraTerm* consiste num terminal que permite a visualização de *outputs* das aplicações, assim como resultados de *debug* de programas executados pelo microprocessador embebido. Por exemplo, o código para impressão no terminal através do comando *fprintf*. Esse terminal permite uma comunicação série com o sistema embebido implementado na FPGA através do módulo *xps_uartlite* já descrito anteriormente, servindo-se da porta UART existente na placa ML605. Durantes os testes há três ligações entre a placa ML605 e um computador:

- Uma conexão entre uma porta USB do computador e a porta JTAG *(Joint Test Action Group)* da placa ML605. Esta conexão constitui o canal de transferência do ficheiro *bitstream* para a FPGA, resultante da implementação do sistema projectado.

- Uma conexão entre uma porta USB do computador e a porta UART da placa ML605. Esta conexão permite a visualização dos resultados da execução de aplicações e resultados de depuração *(debug)* no terminal *TeraTerm*.

- Por último, há a conexão *Ethernet* entre a porta RJ-45 do computador e a porta RJ-45 da placa. É esta que permite testar as duas interfaces *Ethernet* passiveis de serem implementadas na placa ML605.

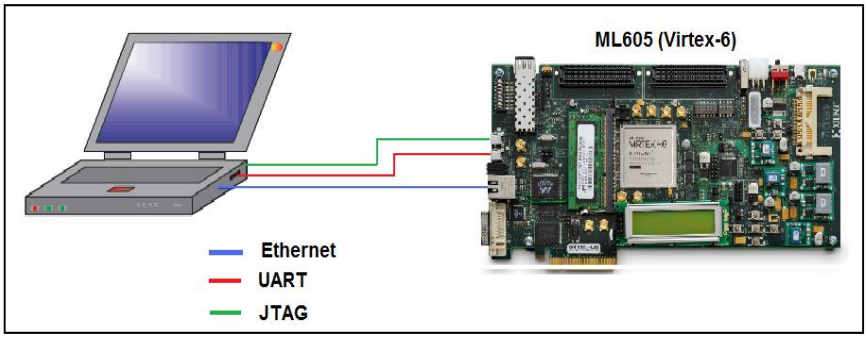


Figura 5.1: *Representação das ligações durante o teste das interfaces Ethernet.*

A figura 5.1 ilustra a montagem de teste, sendo visíveis as 3 conexões entre a placa ML605 e o computador de teste. Nas aplicações de teste, desenvolvidas em C, é atribuído à placa ML605, um endereço IP requerido para o protocolo TCP/IP e um endereço MAC requerido pelo protocolo *Ethernet*. A aplicação fornecida pela *Xilinx* atribui a seguinte configuração *Ethernet* e TCP/IP à placa [26]:

TCP/IP	IP	192.168.1.10
	Máscara de Rede	255.255.255.0
	Gateway	192.168.1.1
Ethernet	Endereço Físico	00.0a.35.00.01.02

Tabela 5.1: *Configuração TCP/IP e Ethernet atribuída à placa ML605.*

O computador de teste foi configurado com um endereço IP conhecido e com a mesma máscara de rede do endereço IP atribuído à placa. O computador foi configurado através das opções disponibilizadas no sistema operativo indicadas na tabela 5.2. Os endereços IP atribuídos à placa e ao computador podem ser alterados nas aplicações.

TCP/IP	IP	192.168.1.100
	Máscara de Rede	255.255.255.0
	Gateway	192.168.1.1

Tabela 5.2: *Configuração TCP/IP atribuída ao computador de teste.*

O terminal *(TeraTerm)* de comunicação em série com o sistema embebido, *TeraTerm* foi configurado com os parâmetros indicados na tabela 5.3.

Baud Rate	9600
Dados	8 *bit*
Paridade	Não
Stop	1 *bit*
Controlo de Fluxo	Não

Tabela 5.3: *Configuração da comunicação em série entre o sistema embebido e o terminal de visualização.*

Quando uma aplicação de teste é executada no microprocessador embebido, são apresentados no terminal os "resultados" da execução da aplicação através da transferência de dados em série utilizando o módulo *xps_uart,* que neste caso consistem em dados sobre os endereços IP, sobre o passo que dever ser efectuado para realizar o teste pretendido e sobre o estado da conexão. A figura 5.2 mostra, no terminal do computador, exemplo do resultado da comunicação série com o sistema embebido, durante a execução de uma das aplicações de teste.

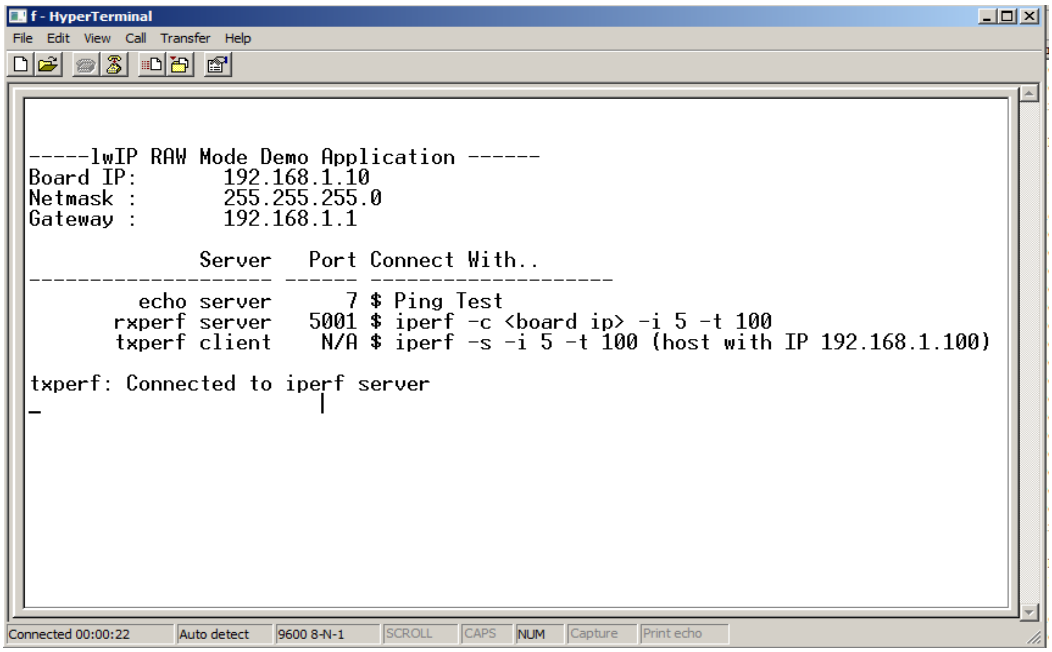


Figura 5.2: *Exemplo da comunicação série entre a placa ML605 e o computador através da porta UART.*

5.1.1 Teste para a Obtenção de Taxas de Transmissão e Recepção de Dados

As aplicações de avaliação de taxas de transmissão permitem determinar a taxa de transmissão máxima de dados utilizando as interfaces *Ethernet* e TCP/IP.

Transmissão

No teste de determinação da taxa de transmissão de dados, uma aplicação *Cliente lwiP* (desenvolvida em C) é executada no microprocessador embebido conectado ao servidor *Iperf* no computador. Este teste consiste no envio contínuo de pacotes de dados de tamanho constante para o computador. A aplicação *Iperf* (no computador) determina a taxa de transmissão de dados e imprime os resultados no terminal de comandos.

Recepção

No teste de determinação da taxa de recepção de dados na interface *Ethernet*, a aplicação lwIP (desenvolvida em C) executada no microprocessador embebido, actua como servidor e aceita os pedidos de conexão realizados pela aplicação *Iperf*, que neste caso funciona como cliente. A aplicação lwIP estabelece a comunicação com o servidor e transmite dados via *Ethernet*. Em intervalos de tempo pré-definidos calcula a quantidade de dados transmitidos e a taxa de transmissão.

5.1.1.1 Resultados

Nesta subsecção são apresentados algumas taxas de transmissão e recepção de dados obtidos com as interfaces *Ethernet* implementadas no sistema embebido.

Interface TMAC

A tabela 5.4 apresenta algumas das taxas de transmissão e recepção de dados obtidas com a interface TMAC, suportando o protocolo TCP/IP, na comunicação entre o PC e a placa, para cada uma das velocidades de ligação possíveis e para cada tipo de API *(Application Programmable Interface)*.

Interface	Velocidade de Ligação	API	Taxa Média	
			Transmissão(Mbps)	Recepção(Mbps)
TMAC	1 Gbps	Raw	101.93	114.92
		Socket	35.56	20.91
	100 Mbps	Raw	53.6	52.67
		Socket	36.55	22.40
	10 Mbps	Raw	9.00	9.00
		Socket	8.47	9.43

Tabela 5.4: *Resultados obtidos para as taxas de transmissão e recepção de dados obtidos utilizando a interface TMAC.*

A partir destes resultados podemos verificar que não foi possível atingir a capacidade máxima de 1 Gbps da interface TMAC indicada pela *Xilinx*. O melhor resultado é obtido quando se utiliza a API no modo *Raw*, havendo taxas de transmissão e recepção acima de 100 Mbps.

Interface ELM

Assim como foi feito para a interface TMAC, são apresentadas na tabela 5.5 as taxas de transmissão e recepção na interface ELM, durante uma comunicação com o PC de teste.

Interface MAC	Velocidade de Ligação	API	Taxas Médias	
			Transmissão (Mbps)	Recepção (Mbps)
Ethernet Lite MAC	100 Mbps	Raw	10.52	12.28
		Socket	16.11	0.69
	10 Mbps	Raw	8.17	8.30
		Socket	3.15	0.53

Tabela 5.5: *Resultados obtidos para as taxas de transmissão e recepção de dados utilizando a interface ELM.*

Na interface EML também não se conseguiu atingir a capacidade máxima de 100 Mbps indicada pela *Xilinx*: apenas 12.28 Mbps na taxa de recepção de dados e 16.11 Mbps na de transmissão.

5.1.2 Teste de Fiabilidade e Acessibilidade

O teste de fiabilidade e de acessibilidade foi realizado com o PING. O teste PING utiliza o protocolo ICMP *(Internet Control Message Protocol)*. No teste PING são enviados à interface alvo pacotes denominados de *echo requests*, ou pacotes ICMP, e espera-se a resposta *(echo reply)*. Neste teste utilizou-se a aplicação *echo server* que responde aos pacotes enviados a partir do computador à placa ML605. O servidor de eco devolve qualquer pacote enviado à placa. No teste PING é calculado o tempo que medeia entre uma transmissão e uma recepção conhecido por *round-trip time*, e são fornecidas também estatísticas de pacotes perdidos na rede. É possível obter a percentagem de pacotes perdidos durante uma comunicação e obter uma estimativa da fiabilidade de comunicação e acessibilidade à interface com recurso ao protocolo ICMP.

5.1.2.1 Resultados

Nesta subsecção são apresentados alguns resultados dos testes de fiabilidade e acessibilidade da interface TMAC à rede, recorrendo ao protocolo TCP/IP.

Interface TMAC

Neste teste, foram enviados 100 pacotes de dados à interface *Ethernet*, esperando que sejam enviados de volta os 100 pacotes, caso a placa ML605 esteja acessível e não haja perdas de pacotes. Na tabela 5.6 encontram-se resultados obtidos com a interface TMAC.

<table>
<tr><th rowspan="2">Interface</th><th rowspan="2">Velocidade de Ligação</th><th rowspan="2">API</th><th colspan="3">Estatística de Pacotes</th></tr>
<tr><th>Pacotes Transmitidos</th><th>Pacotes Recebidos</th><th>% Pacotes Perdidos</th></tr>
<tr><td rowspan="6">TMAC</td><td rowspan="2">1 Gbps</td><td>Raw</td><td rowspan="6">100</td><td rowspan="6">100</td><td rowspan="6">0</td></tr>
<tr><td>Socket</td></tr>
<tr><td rowspan="2">100 Mbps</td><td>Raw</td></tr>
<tr><td>Socket</td></tr>
<tr><td rowspan="2">10 Mbps</td><td>Raw</td></tr>
<tr><td>Socket</td></tr>
</table>

Tabela 5.6: *Resultados do teste de fiabilidade e acessibilidade da interface TMAC utilizando o teste PING.*

Em todos os testes PING efectuados não houve perda de pacotes, o que nos dá uma ideia de uma boa fiabilidade de comunicação e acessibilidade na interface TMAC.

Interface ELM

Na tabela 5.7 encontra-se alguns dos resultados obtidos, quando 100 pacotes foram enviados à placa com a interface ELM.

Interface MAC	Velocidade de Ligação	API	Estatística de Pacotes		
			Pacotes Transmitidos	Pacotes recebidos	% Pacotes Perdidos
Ethernet Lite MAC	100 Mbps	Raw	100	100	0
		Socket			
	10 Mbps	Raw			
		Socket			

Tabela 5.7: *Resultado do teste de fiabilidade e acessibilidade da interface ELM utilizando o teste PING.*

Neste caso, podemos concluir que a interface ELM é bastante fiável para a comunicação e apresenta boa acessibilidade, dado que não houve perda de pacotes durante todos os testes PING realizados.

5.1.3 Controlo da Placa via *Ethernet*

A *Xilinx* disponibiliza um servidor *Web* baseado em TCP/IP, que permite controlar e monitorizar componentes de *hardware* da placa ML605, nomeadamente controlar o estado dos LEDs e monitorizar o estado dos interruptores DIP *(Dual In-line Package)* da placa ML605. Este *servidor Web,* ao ser executado pelo sistema embebido permite controlar e monitorizar a placa ML605 a partir do computador, via *Ethernet*, bastando apenas ser conhecido o endereço IP do servidor e ter um *browser* (figura 5.3) instalado no computador. Este servidor *Web* foi utilizado para testar o controlo da placa ML605 via *Ethernet* (utilizando a interface TMAC, a interface escolhida para o testador).

Pelo endereço *http://192.168.1.10* (endereço IP da placa, atribuído ao servidor *Web*) acede-se ao servidor *Web* executado pelo sistema embebido a partir do computador de teste, e isso permite enviar comandos para acender ou apagar os LEDs da placa e permite também enviar comados para ler o registo de estado do Interruptores DIP da placa ML605, via *Ethernet*. Esta é uma situação similar à implementada no testador MobiDICK 4, onde são enviados comandos para um sistema embebido para a realização de testes electrónicos, via *Ethernet*.

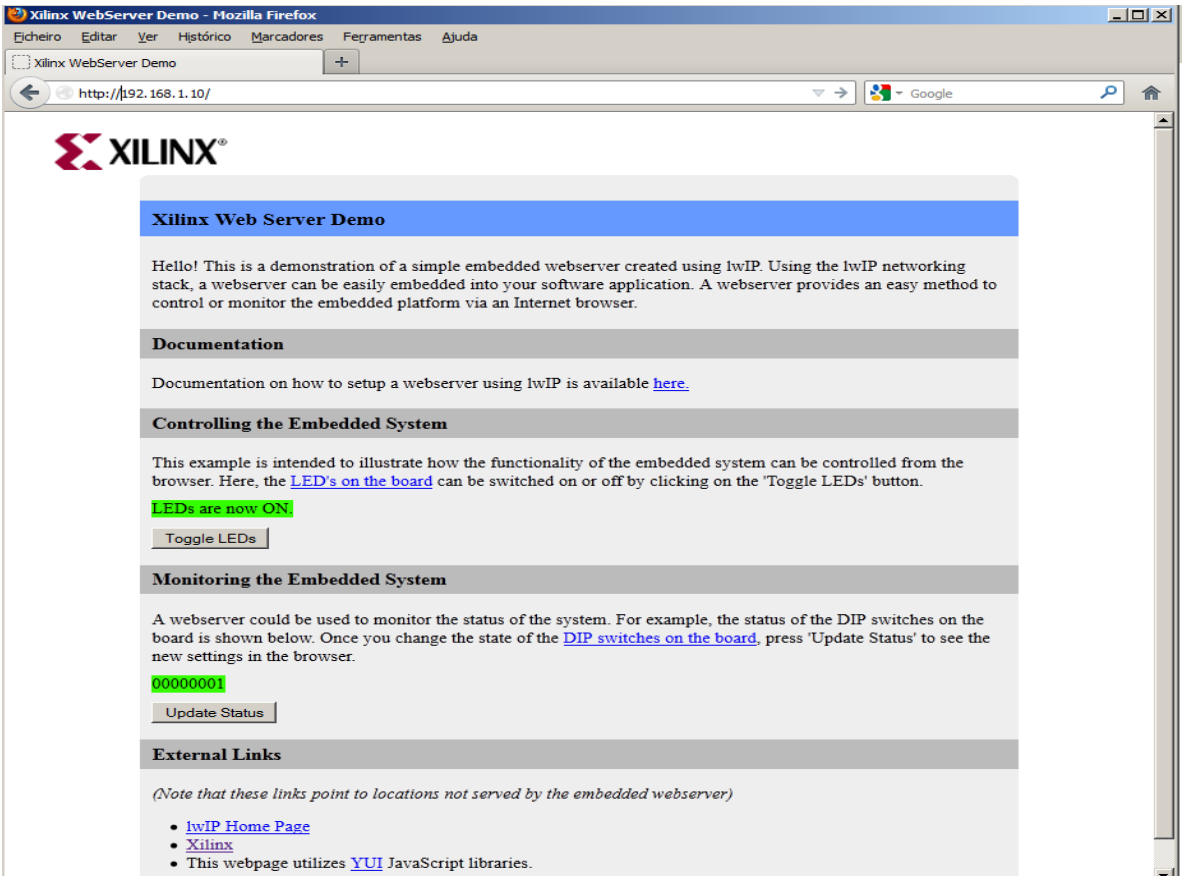


Figura 5.3: *Interface de comunicação com um servidor web executado no sistema embebido.*

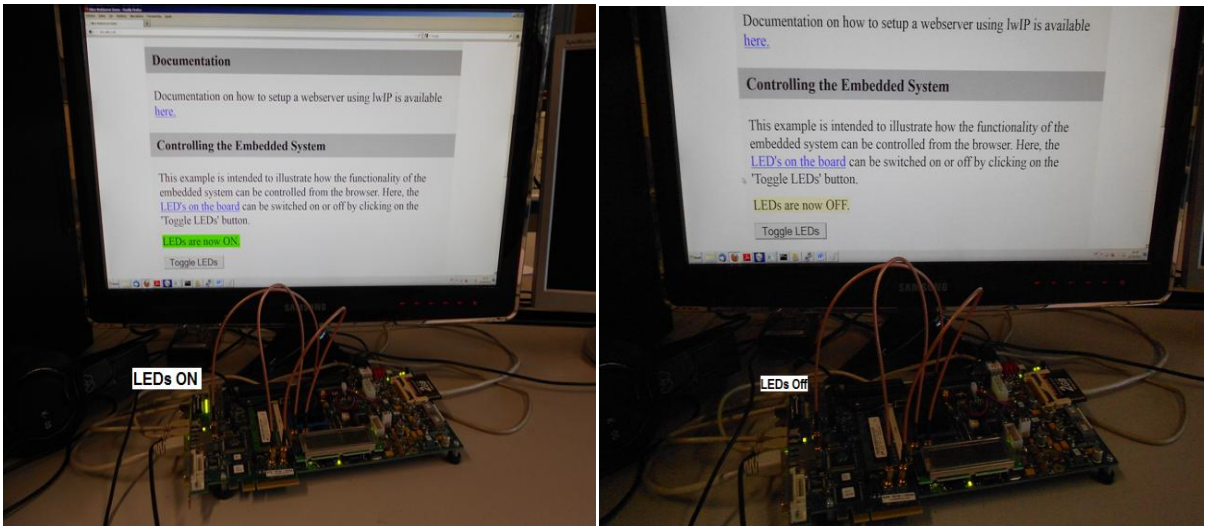


Figura 5.4: *Controlo da Placa ML605 via Ethernet.*

A figura 5.4 mostra o controlo dos LEDs da placa a partir de um computador, via *Ethernet* (TMAC). A partir dos resultados obtidos em todos os testes podemos concluir que a interface *Ethernet* TMAC é completamente funcional, oferece melhor desempenho em relação à interface ELM, e por estas razões recomendou-se a sua implementação no MobiDICK 4 para suportar a comunicação com a aplicação *Willy*.

5.2 Simulação e Testes do Módulo CRC

5.2.1 Simulação

O módulo CRC foi simulado na ferramenta ISE da *Xilinx*, em concreto no simulador de circuitos digitais ISIM. Foi desenvolvido um *test bench* para a simulação, onde foi utilizado como estímulo um pacote de dados enviados pela Placa de Interface numa situação real (figura 5.5).

1	0x51115110	52	0x42c08c33		
2	0x981c76cf	53	0x42d09033		
3	0x42c0cc23	54	0x02c08433		
4	0x42c0c424	55	0x00b4f08d		
5	0x02a0c825	56	0x981c76cf		
6	0x02c0c426	57	0x42a08426	103	0x03e0ec2a
7	0x42c0c424	58	0x42908425	104	0x43d0e82a
8	0x42d0d422	59	0x42a08425	105	0x43b0e82a
9	0x42d0cc22	60	0x02808426	106	0x43a0ec2a
10	0x4f29cc03	61	0x02808425	107	0x0390e82a
11	0x981c76cf	62	0x02808426	108	0x03a0e82a
12	0x01a0bc23	63	0x02808426	109	0x6e512c29
13	0x0180b825	64	0x54732070	110	0x981c76cf
14	0x41a0c023	65	0x981c76cf	111	0x4380ac2a
15	0x41c0b825	66	0x4340a839	112	0x4370b42c
16	0x01b0bc22	67	0x4310b43b	113	0x0380a42a
17	0x4190bc24	68	0x4310b036	114	0x0390a42b
18	0x0190bc23	69	0x0310b038	115	0x4380b02e
19	0x7417d398	70	0x0300ac38	116	0x4390a82c
20	0x981c76cf	71	0x0320a838	117	0x4390b42d
21	0x02408c2d	72	0x42f0ac37	118	0x764653af
22	0x4230942b	73	0x61e63190	119	0x981c76cf
23	0x02408c2d	74	0x00000000	120	0x02d0b825
24	0x0240882c	75	0x00000000	121	0x42e0c027
25	0x4240902b	76	0x00000000	122	0x42c0b426
26	0x02408c2d	77	0x00000000	123	0x42d0b828
27	0x4270902b	78	0x00000000	124	0x02e0b429
28	0x7f691a93	79	0x00000000	125	0x42f0ac25
29	0x981c76cf	80	0x00000000	126	0x02e0b829
30	0x4390c02d	81	0x00000000	127	0x079b341b
31	0x0390c428	82	0x00000000	128	0x00000000
32	0x0390c02f	83	0x00000000	129	0x00000000
33	0x0390c02c	84	0x00000000	130	0x00000000
34	0x4390c02e	85	0x00000000	131	0x00000000
35	0x03d0c02d	86	0x00000000	132	0x00000000
36	0x0330c42e	87	0x00000000	133	0x00000000
37	0x34e5ce61	88	0x00000000	134	0x00000000
38	0x981c76cf	89	0x00000000	135	0x00000000
39	0x4330a826	90	0x00000000	136	0x00000000
40	0x4350ac24	91	0x00000000	137	0x00000000
41	0x4330a024	92	0x981c76cf	138	0x00000000
42	0x0340a825	93	0x4290ec28	139	0x00000000
43	0x4330a423	94	0x02a0f027	140	0x00000000
44	0x0320a425	95	0x42c0f029	141	0x00000000
45	0x4320a824	96	0x42a0f029	142	0x00000000
46	0x26911832	97	0x42b0e828	143	0x00000000
47	0x981c76cf	98	0x0290ec26	144	0x00000000
48	0x02d09034	99	0x02b0e426	145	0x00000000
49	0x02a08830	100	0x2e41e3f9	146	0x0fff0fff
50	0x02a09432	101	0x981c76cf	147	0x00005779
51	0x02b0882f	102	0x0390e82a	148	0x3fff3ff0

Figura 5.5: *Pacote de dados utilizado como estímulo durante a simulação do algoritmo CRC.*

A figura 5.5 ilustra o pacote de dados utilizado como sinal de estímulo nas simulações do módulo CRC. Este pacote foi adquirido numa situação real de transmissão correcta entre as electrónicas de frontaria e de retaguarda. A primeira palavra digital *0x51115110* corresponde ao cabeçalho global, a segunda palavra digital *0x981c76cf*, corresponde ao cabeçalho do primeiro DMU, que é comum a todos os restantes DMUs, seguido de 7 palavras digitais correspondentes às amostras seguido do valor de CRC do respectivo DMU, que para o primeiro DMU apresenta o valor *0x4f29cc03* (*0x4f29* para *bits* par e 0xcc03 para *bits* ímpar). Este padrão repete-se até á palavra digital *0x0fff0fff,* que corresponde à palavra digital *Dmu_chip_mask*. De seguida, temos uma palavra digital com os primeiros 16 *bits* mais significativos nulos, e os 16 *bits* menos significativos com o valor *0x5779* que corresponde ao valor do *CRC Global*. No sistema ROD é calculado o valor do *CRC Global* para ser inserido na parte dos 16 *bits* mais significativos nulos; no entanto, no MobiDICK 4 esse processo não é realizado, sendo apenas utilizados os 16 *bits* menos significativos para a verificação da integridade dos dados.

5.2.1.1 Resultados de Simulação

O pacote de dados utilizado não apresenta erros nos CRCs. Foram realizadas simulações para duas situações:

1. Utilizando o pacote de dados com valores correctos, isto é, assim como foi enviado pela Placa de Interface da electrónica de frontaria. Nesta situação espera-se que o algoritmo calcule os valores correctos de CRC, que deverão ser iguais aos anexados no pacote de dados. Neste caso, em que nenhum erro é detectado, os contadores permanecerão com o valor zero.

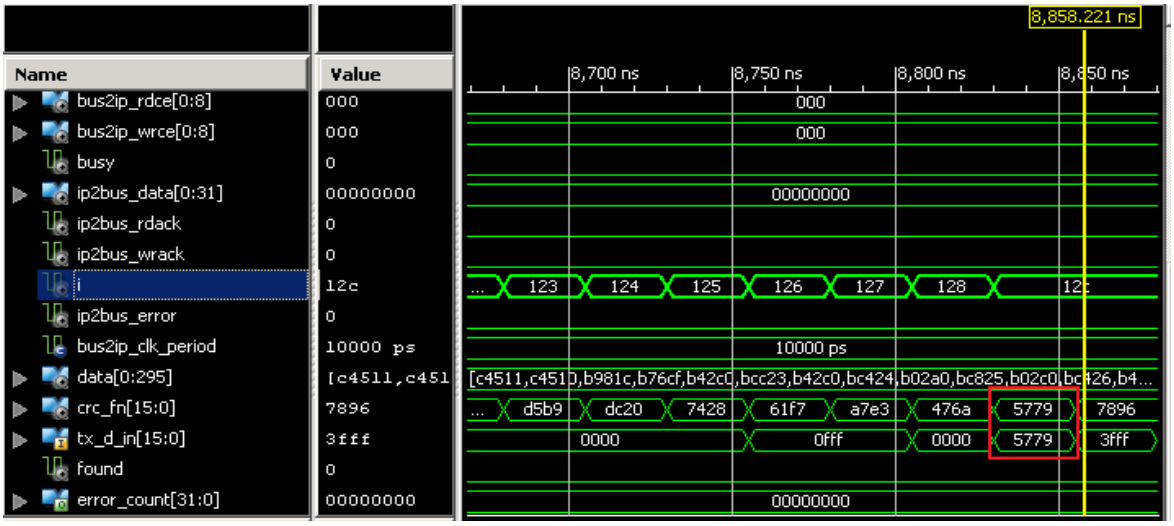


Figura 5.6: *Resultado de simulação para o valor correcto de CRC Global.*

A figura 5.6 mostra um dos resultados obtidos para o valor de *CRC Global*. Como era esperado, o módulo CRC calculou um valor de *CRC Global* exactamente igual ao valor anexado no pacote de dados utilizado como estímulo, correspondendo a *0x5779*. Verifica-se também que o valor do contador *(error_count)* permaneceu igual a zero.

A mesma situação ocorre para os valores de CRC dos pacotes de dados de cada DMU, como se pode verificar na figura 5.7. Neste caso, são apresentados apenas os valores para o primeiro pacote DMU, mas verifica-se que, para todos os DMUs, o módulo CRC calcula os CRCs correctos para *bits* pares como para os *bits* ímpares. Para este pacote DMU em particular, os CRCs anexados no pacote de estímulo *(tx_d_in)* são *0x4f29* (*bits* pares) e *0xcc03* (*bits* ímpares) que são exactamente iguais aos valores calculados, isto é, os valores dos sinais *newcrc1* (*bits* pares) e *newcrc2* (*bits* ímpares) indicados num rectângulo vermelho na parte inferior da figura 5.7.

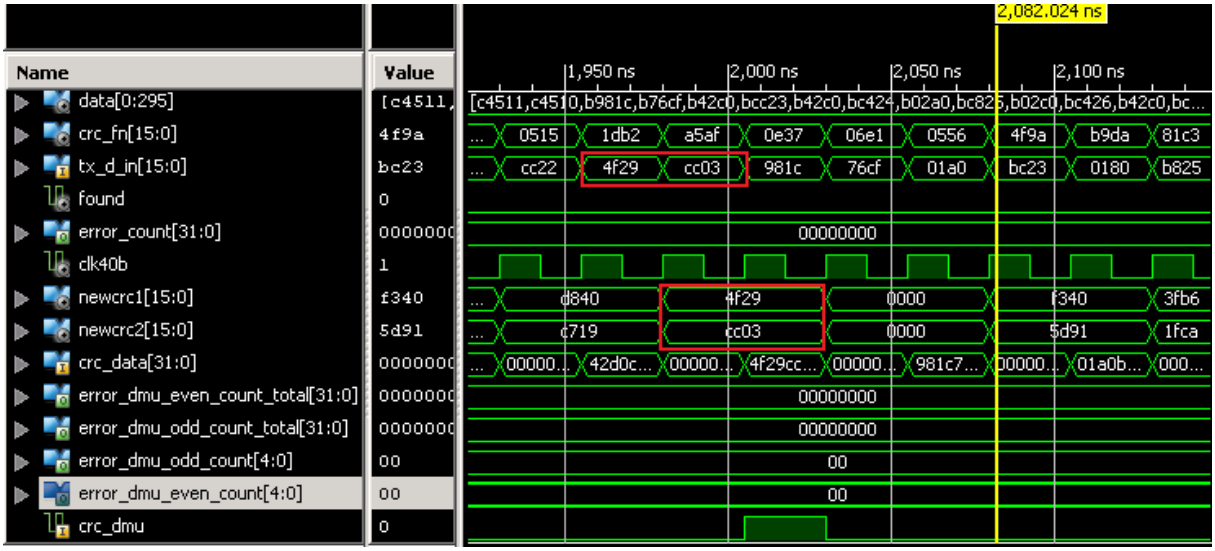


Figura 5.7: *Resultado de simulação para os valores correctos de CRC dos DMUs.*

2. Corrompendo o anterior pacote de dados, isto é, alterando alguns dos seus *bits*. Com a alteração de *bits* no pacote de dados, o módulo CRC calculará CRCs diferentes daqueles anexados ao pacote de dados de cada DMU, e logo detectará e contabilizará os respectivos erros.

Para esta situação, o módulo CRC comportou-se conforme o esperado. Calculou um CRC diferente do valor de *CRC Global* anexado ao pacote de estímulo (figura 5.8). O valor de *CRC Global* anexado ao pacote é *0x5779*, ao passo que o valor calculado foi *0x6d7a*. Nesta situação de erro é activado o sinal *found* com o valor '1', e este sinal é por sua vez detectado pelo contador de erros, que incrementa o seu valor de uma unidade (rectângulo azul).

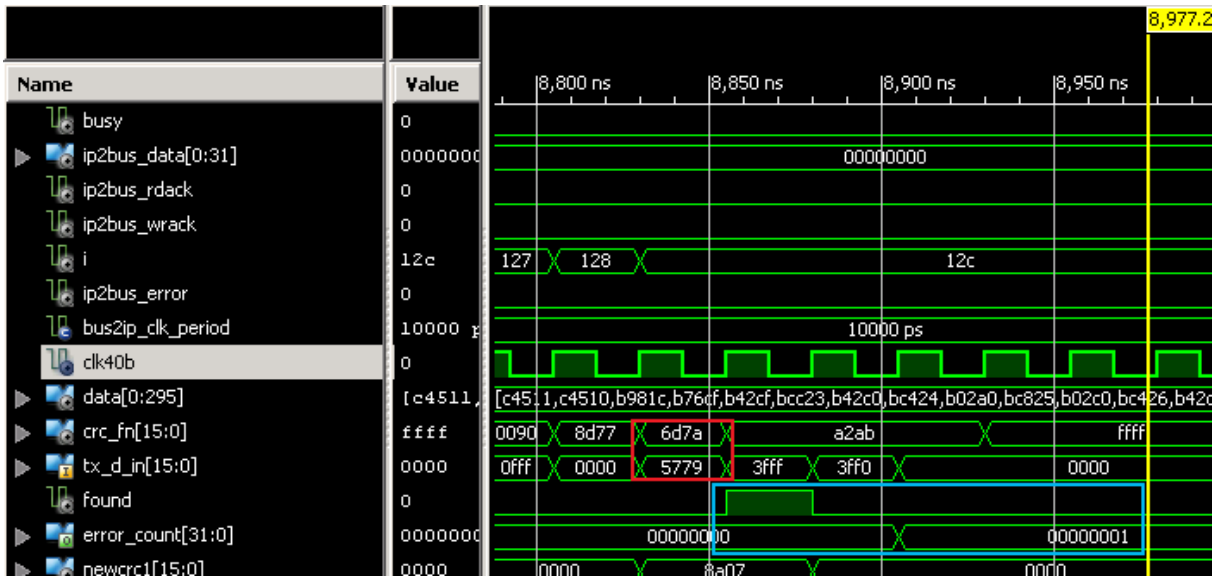


Figura 5.8: *Resultado de simulação para o valor incorrecto de CRC Global.*

A corrupção dos dados foi efectuada no primeiro pacote de dados do primeiro DMU. A figura 5.9 ilustra o comportamento em relação aos valores de CRC do pacote DMU corrompido. Como se pode verificar, o módulo CRC calculou valores de CRC diferentes dos valores anexados no pacote de estímulo. Os valores anexados ao primeiro pacote de dados do primeiro DMU são, como já se referiu, *0x4f29* (*bits* pares) e *0xcc03* (*bits* ímpares) no sinal *tx_d_in*, os valores calculados, neste caso, são *0xefea* (*bits* pares) e *0x6cc0* (*bits* ímpares). Verifica-se também o correcto funcionamento dos contadores, que foram incrementados ao detectarem os erros nos CRCs, assinalados com um rectângulo azul na figura 5.9.

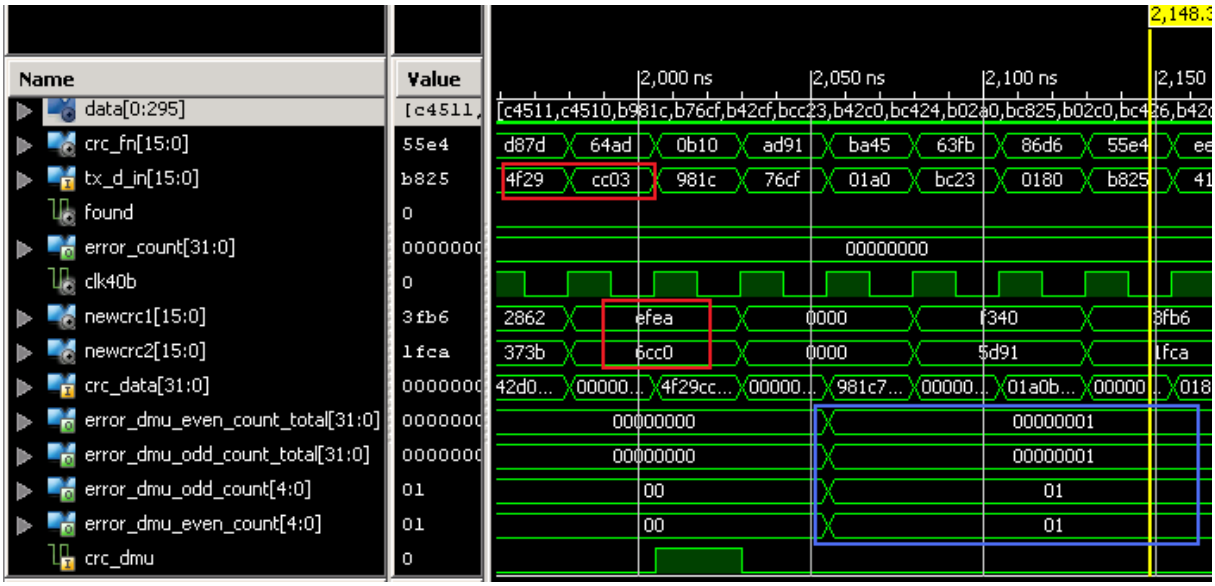


Figura 5.9: *Resultado de simulação para os valores incorrectos de CRC dos DMUs.*

5.2.2 Teste

Para realizar o teste do módulo CRC foi disponibilizada, pela equipa da Universidade de Valência, uma aplicação desenvolvida em C++ chamada de *Test-stress* e implementada por eles, com instruções para serem executadas pelo microprocessador embebido, o *PowerPC*. Esta aplicação permite configurar a electrónica de frontaria, enviar comandos L1A solicitando à electrónica de frontaria o envio de dados, configurar o módulo *G-Link* para receber dados, ler os registos dos contadores do módulo CRC e apresentar os resultados.

O emulador *G-Link* do MobiDICK 4, já com o módulo CRC integrado realiza a verificação da integridade dos dados enviados pela electrónica de frontaria, realiza a contagens dos erros, e actualiza os registos dos contadores, os resultados são lidos pelo microprocessador e apresentados ao utilizador.

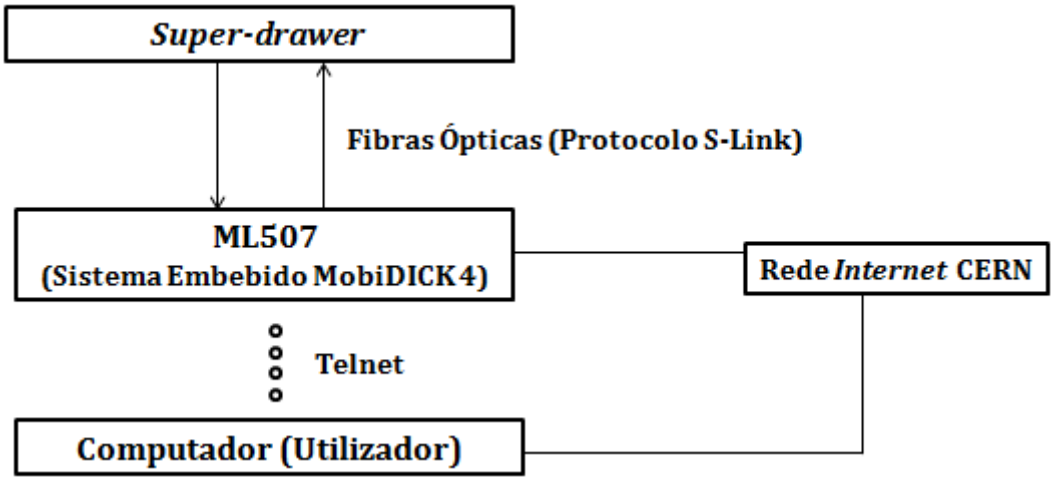


Figura 5.10: *Esquema de teste utilizado para a validação do módulo CRC.*

A figura 5.10 ilustra o esquema utilizado para a validação do módulo CRC. Durante os testes do módulo CRC, a placa ML507, carregada com o sistema embebido do MobiDICK 4, encontra-se conectada por fibras ópticas, que permitem enviar comandos TTC e receber dados, a um *Super-drawer* de teste instalado no laboratório do CERN. O MobiDICK 4 é ligado à rede *Internet* do CERN, o que permite que um computador ligado à mesma rede interaja com ele por *Telnet,* uma aplicação TCP/IP disponível no sistema operativo do computador utilizado.

Para além da aplicação *Test-stress* executada pelo microprocessador foi utilizada a ferramenta *Chipscope Analyzer* da *Xilinx* para a validação do módulo CRC. A ferramenta *Chipscope Analyzer* permite monitorizar os sinais do sistema implementado na FPGA, e neste caso foi utilizada para verificar se o funcionamento do módulo CRC, isto é, para verificar se este calculava os CRCs correctamente.

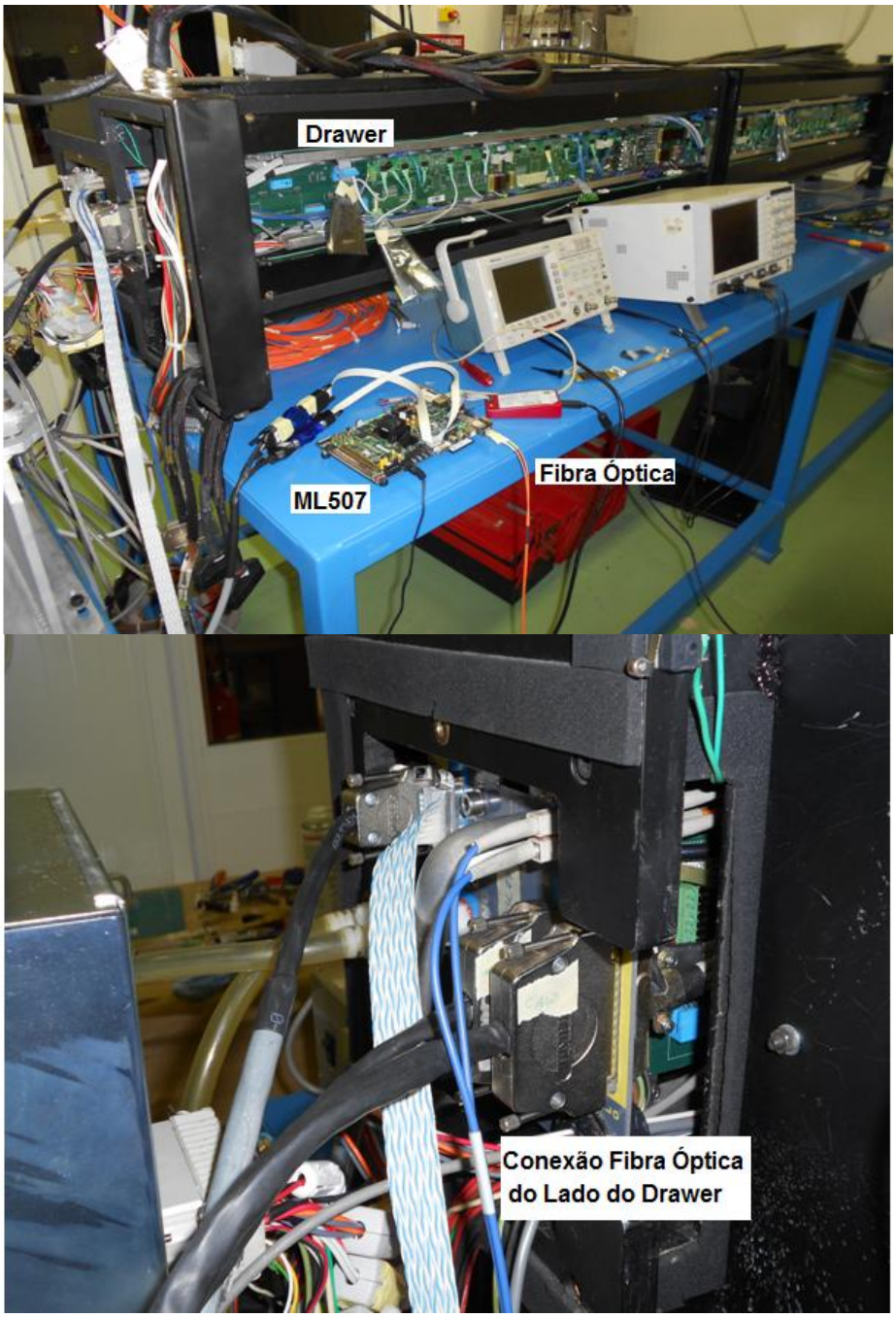


Figura 5.11: *Bancada de teste para a validação do módulo CRC.*

A figura 5.11 é uma fotografia da bancada de teste montada num dos laboratórios do CERN para a validação do sistema embebido do MobiDICK 4, particularmente para a validação do módulo CRC.

5.2.2.1 Resultados

Nesta secção são apresentados os resultados do teste e validação do módulo CRC implementado no testador MobiDICK 4. Utilizou-se a ferramenta *Chipscope Analyzer* para monitorizar os sinais de interesse na FPGA. Estes resultados permitiram-nos verificar se o módulo CRC se comportava de acordo com as simulações.

CRC Global

A figura 5.12 mostra um dos resultados obtidos no teste do módulo CRC. Este teste foi realizado com o objectivo de verificar se aquele módulo calculava correctamente o valor de *CRC Global* numa situação real de aquisição de dados vindos da electrónica de frontaria.

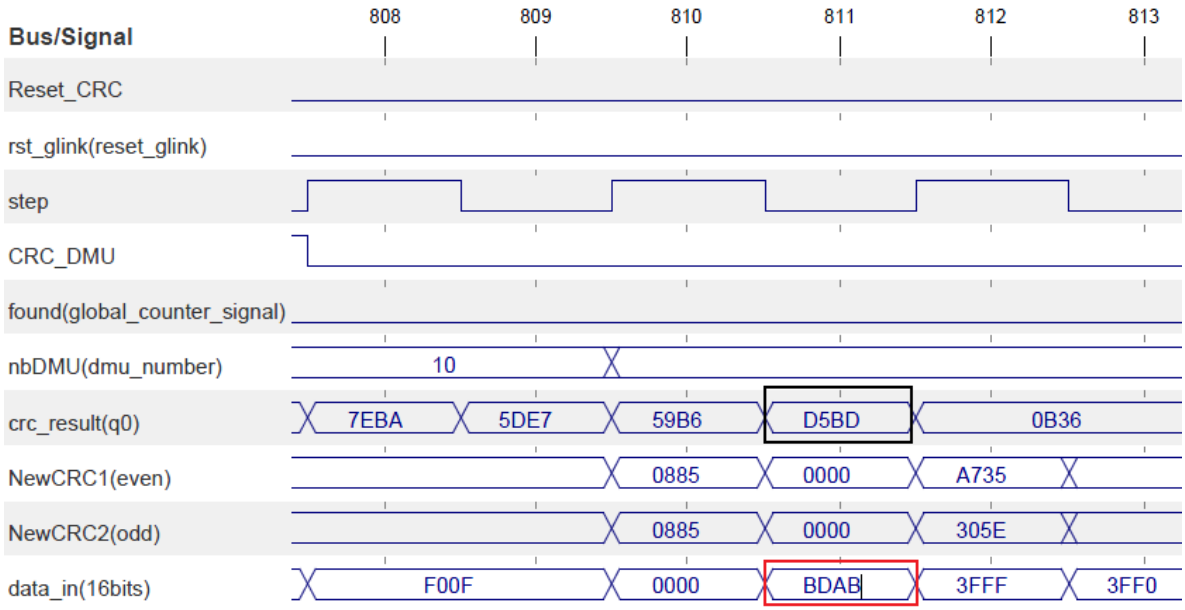


Figura 5.12: *Resultado obtido no teste e validação do cálculo do valor de CRC Global.*

Neste caso, adquiriu-se um pacote não corrompido pelo ruído, com um valor de *CRC Global* igual *0xBDAB*, assinalado com um rectângulo vermelho na figura 5.12. O *CRC Global* calculado encontra-se assinalado com um rectângulo de cor preta, e apresenta o valor hexadecimal *0xD5BD*. O módulo CRC efectua a inversão da ordem dos *bits* do valor de CRC calculado antes de compará-lo com o valor anexado. Portanto, se invertermos a ordem dos *bits* da palavra digital *0xD5DB (crc_result)*, começando pelo *bit* menos significativo, obtém-se a palavra digital *0xBDAB*, que é exactamente igual ao *CRC Global* anexado ao pacote enviado pela Placa de Interface da electrónica de frontaria. A partir deste resultado conclui-se que o módulo CRC calcula correctamente o *CRC Global*.

CRC de DMUs

A figura 5.13 mostra um dos resultados obtidos no teste e validação do cálculo de CRC para os pacotes de dados dos DMUs. Nesta situação adquiriu-se um pacote de dados e verificou-se se o módulo CRC calcula correctamente os valores de CRC associado aos *bits* pares e ímpares das palavras digitais enviados pela Placa de Interface da electrónica de frontaria.

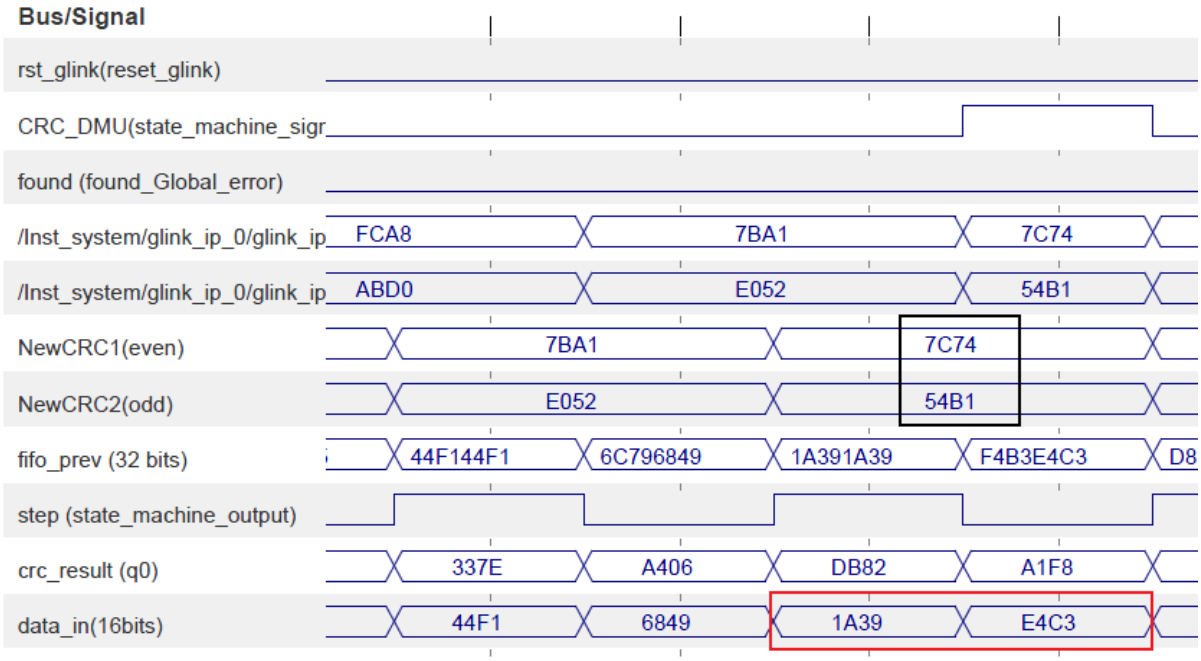


Figura 5.13: *Resultado obtido no teste e validação do cálculo dos valores de CRC associados aos pacotes de cada DMU.*

Os primeiros testes mostraram que o módulo CRC não se comportou como seria de esperar. Para este pacote de dados da figura 5.13 em especifico, o módulo CRC não foi capaz de calcular os valores de CRC correctamente. Para a situação em particular, os valores de CRC anexados ao pacote DMU são *0x1A39* (*bits* pares) e *0xE4C3* (*bits* ímpares), ambos assinalados com um rectângulo vermelho, ao passo que, os valores calculados são *0x7C74* (*bits* pares) e *0x54B1* (*bits* ímpares). O motivo pelo qual esta situação acontece foi identificado, e concluiu-se que não é um problema no algoritmo de cálculo dos valores de CRC, mas sim um problema na organização das palavras digitais de 16 *bits* em palavras digitais de 32 *bits*, uma funcionalidade implementada no módulo *DMU_crc*. Esta organização não estava a ser efectuada correctamente, isto é, dados incorrectos estavam a ser passados ao módulo *crcDMU* responsável pelo cálculo dos valores de CRC, induzindo-o a calcular valores de CRC complemente incorrectos.

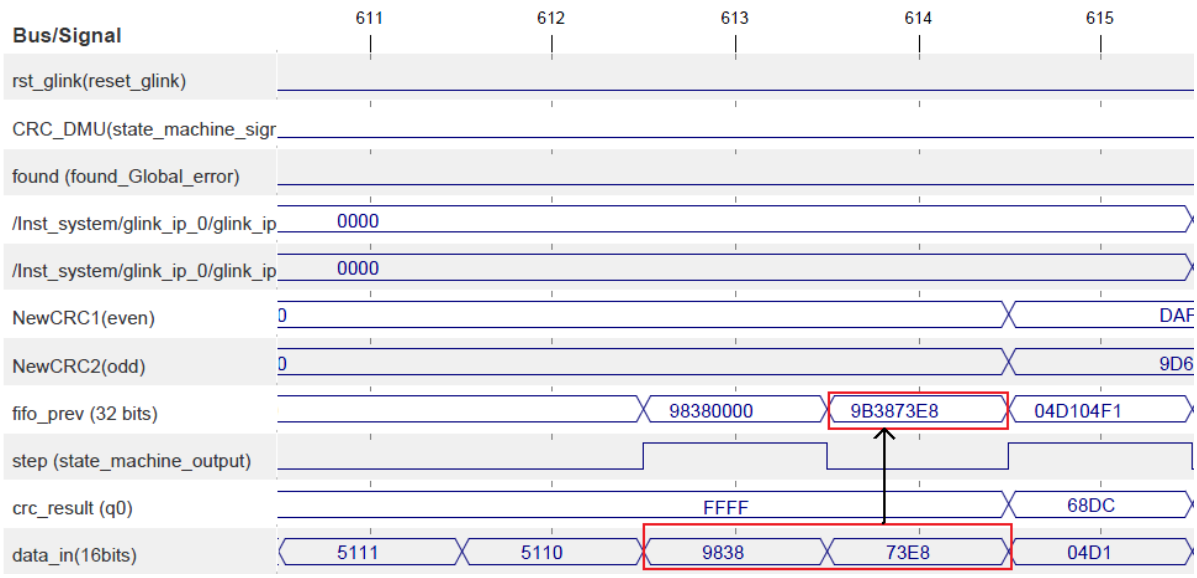


Figura 5.14: *Organização incorrecta de dados pelo módulo CRC.*

A figura 5.14 mostra a referida organização incorrecta dos dados. Tal como já foi referido, a electrónica de frontaria envia dados em palavras digitais de 16 *bits* (sinal *data_in*) e cada par de duas palavras digitais de 16 *bits*, forma uma única palavra digital de 32 *bits*, no caso assinalado a vermelho no sinal *data_in*, as duas palavras digitais de 16 *bits 0x9838* e 0x73E8 juntos formam uma única palavra digital de 32 *bits* igual a *0x983873E8* (corresponde ao cabeçalho do primeiro pacote DMU). A reorganização é realizada no registo *fifo_prev*, quando o sinal *step* apresentar o valor '0'. O sinal *step* representa um sinal de saída da máquina de estados descrito anteriormente, sendo utilizado como sinal de controlo para o processo de organização dos dados. Espera-se, neste caso, quando *step = '0'*, que o registo *fifo_prev* apresente o cabeçalho já organizado formando uma única palavra digital igual a *0x983873E8*, e que suceda o mesmo para as outras palavras digitais. No entanto, verificou-se uma situação inesperada, como se pode verificar na figura 5.15, o registo *fifo_prev* apresenta o valor *0x9B3873E8*, o qual apresenta o segundo *byte* corrompido. Este resultado é totalmente diferente dos resultados obtidos nas simulações.

Uma nova versão que apresenta uma correcção a esta situação foi desenvolvida, no entanto, não foi testada a tempo de apresentar novos resultados neste documento que indicassem que a situação foi solucionada. Isto deve-se ao facto do sistema MobiDICK 4 apresentar vários componentes e vários grupos que colaboram na implementação do testador MobiDICK 4 e portanto, o sistema foi entretanto disponibilizado a outro grupo para testes de outros componentes.

Aplicação *Test-stress*

A aplicação *Test-stress* executada pelo *PowerPC* embebido, permite configurar a electrónica de frontaria e enviar o comando L1A a solicitar à electrónica de frontaria o envio de pacote de dados. O emulador *G-Link*, ao receber esses dados, realiza a contagens de eventos (pacotes de dados), a verificação da integridade de dados (através do cálculo dos CRCs), realiza a contagem de erros e actualiza os contadores. Aquela aplicação permite que o microprocessador leia os registos do módulo *G-Link* e apresente os resultados.

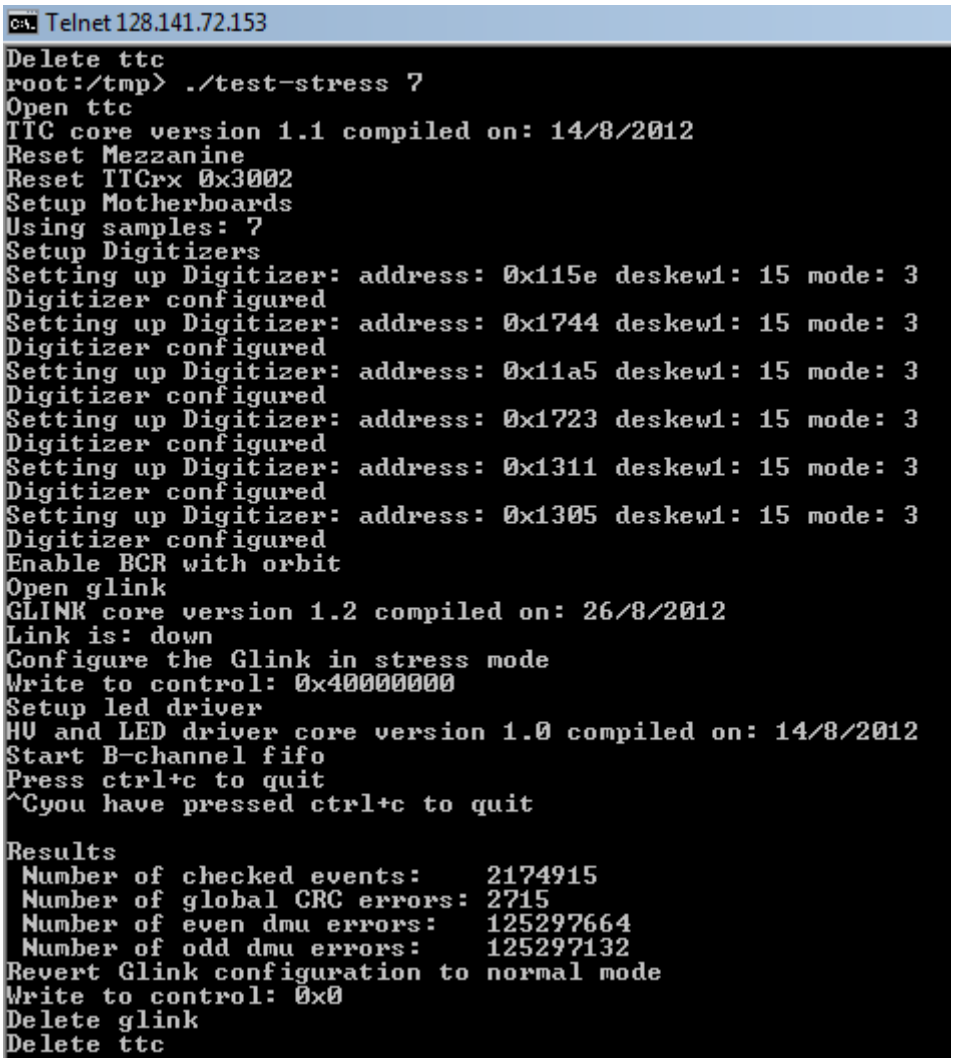


Figura 5.15: *Resultado obtido a partir da leitura dos registos do emulador G-Link.*

A figura 5.15 mostra resultados obtidos com a aplicação de teste *Test-stress*, relativos a estatísticas de pacotes e de erros. Para esta situação em particular contabilizou-se 2174915 eventos ou pacotes de dados, dos quais 2715 eventos apresentam erros associados ao valor de *CRC Global*, o que representa uma percentagem de pacotes com erros de cerca de 13 %

relativamente ao número de pacotes contabilizados. Para os outros testes a percentagem de pacotes com erros encontra-se na gama de 13 a 15 %, um resultado dentro das expectativas para o *Super-drawer* de teste do laboratório do CERN.

Para além do cálculo de CRC de pacotes de cada DMU não estar a ser efectuado correctamente, verificou-se também que os contadores associados aos valores de CRC de DMUs, não estavam a funcionar correctamente, em total desacordo com os resultados obtidos por simulação. Nesta situação em particular, o valor dos registos dos contadores apresentaram os valores: 125297664 (contagens de erros associados a CRC *bits* pares) e 125297132 (contagem de erros associados a CRC *bits* ímpares). Esses valores são superiores a número de eventos total de pacotes DMUs adquiridos, uma situação anormal, dado que não pode haver contagens de erros de valor superior ao número de eventos. Pensamos que esta contagem incorrecta esteja relacionada como facto de que o módulo CRC não calcular os valores de CRC correctamente. Espera-se que, a nova versão já desenvolvida, mas não testada ainda, calcule valores correctos de CRC e realize contagens correctas de erros nos pacotes de dados de DMUs.

5.3 Conclusões

Neste capítulo fez-se a descrição dos testes efectuados e os resultados obtidos no teste das interfaces *Ethernet* e na simulação e teste do módulo CRC implementado no testador MobiDICK 4.

Foram realizadas testes de comunicação utilizando as interfaces *Ethernet* implementadas com recurso ao protocolo TCP/IP. Foram realizados testes que permitiram determinar o desempenho da comunicação, através da obtenção de taxas de transmissão e recepção de dados, testes de fiabilidade e acessibilidade utilizando o teste PING. Foi também realizado uma demonstração de controlo e monitorização da placa ML605, via *Ethernet* (módulo TMAC) a partir do computador.

O módulo CRC foi simulado utilizando o simulador ISIM da *Xilinx* e o seu teste foi realizado numa situação real de aquisição de dados da electrónica de frontaria. Os resultados de simulações indicam um correcto funcionamento do módulo CRC, ao passo que os resultados dos testes mostram que aquele funciona correctamente apenas de forma parcial. O problema foi identificado e já foi desenvolvida que será testada num futuro próximo.

Capítulo 6 Conclusões

Neste trabalho foram implementadas e testadas duas interfaces *Ethernet* e um módulo de verificação de integridade de dados baseado no algoritmo CRC, no âmbito da actualização de um testador de sistemas electrónicos do calorímetro hadrónico *TileCal* da experiência ATLAS/CERN.

Este testador é conhecido como MobiDICK 4, e é utilizado para testar a funcionalidade da electrónica de frontaria do *TileCal*. Este sistema comunica com o utilizador via *Ethernet*, pelo que foi necessário proceder a testes de comunicação utilizando as interfaces disponíveis para implementação em FPGAs, de forma a verificar as suas fiabilidade e desempenho na comunicação. Uma da funcionalidades do testador MobiDICK 4 é verificar a integridade de dados enviados pela electrónica de frontaria, por isso foi implementado e testado um módulo capaz de realizar esta tarefa, baseando-se no algoritmo habitualmente conhecido por *Cyclic Redundancy Check*. Este algoritmo já se encontra implementado no *TileCal* para a verificação da integridade de dados transmitidos da electrónica de frontaria para a electrónica de retaguarda. Durante a realização de testes, a electrónica de frontaria comunica com o referido testador MobiDICK 4, e consequentemente o mesmo módulo de verificação de integridades de dados implementado utilizado na electrónica de retaguarda foi implementado no testador.

No capítulo 1, efectou-se uma introdução, incluindo a descrição da plataforma da *Xilinx* para o desenvolvimento de sistema embebidos que foi utilizada para a implementação e para os testes de interfaces *Ethernet*, a plataforma *Embedded Development Kitt* (EDK). A ferramenta EDK é uma ferramenta, que permite o desenvolvimento de sistemas embebidos completamente funcionais visando a implementação em FPGAs da *Xilinx*. É também realizada uma breve descrição da tecnologia FPGA, dos sistemas embebidos e da experiência ATLAS do CERN.

O capítulo 2 foi dedicado à descrição do calorímetro hadrónico *TileCal*, da electrónica de frontaria instalada nesse calorímetro e do testador MobiDICK. Estudou-se a estrutura do calorímetro e a electrónica associada à aquisição do sinal produzido pelas partículas que atravessam o detector. Esse estudo permitiu compreender melhor o testador, a sua arquictetura e os testes à electrónica de frontaria efectuados pelo MobiDICK.

O capítulo 3 é dedicado à descrição do sistema dedicado ao teste das duas interfaces *Ethernet*: *Tri-Mode Media Access Controller* (TMAC) e *Ethernet Lite Media Access Controller*

(ELM). É apresentada a descrição dos principais componentes de *hardware* do sistema embebido implementado, assim como a descrição das aplicações de *software* utilizadas no teste. A interface ELM apresenta uma limitação óbvia em relação à interface TMAC, isto é, não suporta a capacidade de ligação de 1 Gbps, mas realiza as mesmas funções com a vantagem de ocupar menos recursos na FPGA, de acordo com as indicações da *Xilinx*. Neste caso usou-se como critério preferencial o desempenho, em detrimento da gestão de recursos na FPGA, para a escolha da interface *Ethernet* adequada à implementação no testador, ou seja, foi escolhida para a implementação final a interface TMAC.

O capítulo 4 é dedicado à descrição da implementação do módulo de verificação da integridade de dados no testador MobiDICK 4. O módulo implementado é capaz de verificar se a transmissão de dados entre a electrónica de frontaria e o testador ocorre de forma correcta ou não, ao mesmo tempo contabilizar os erros. O processo de verificação de integridade de dados implementado baseia-se no algoritmo CRC. Para além da descrição deste, é também apresentado o processo de verificação da integridade de dados implementado no *TileCal*.

O capítulo 5 foi dedicado à descrição de testes efectuados e à apresentação dos seus resultados. Neste capítulo descreveram-se os testes de comunicação entre a placa ML605 e um computador, utilizando as interfaces *Ethernet* implementadas com recurso ao protocolo TCP/IP, situação similar implementado no testador MobiDICK 4. Foram também descritas as simulações e os testes efectuados ao módulo CRC numa situação real de aquisição de dados da electrónica de frontaria.

A partir dos resultados obtidos no teste das interfaces *Ethernet* concluiu-se que a interface TMAC apresentava melhor desempenho do que a interface ELM, relativamente às taxas de transmissão e recepção de dados numa comunicação com um computador de teste (tabelas 5.4 e 5.5). Tomando uma situação concreta como exemplo, no caso em que a velocidade de ligação é de 100 Mbps, no modo *Socket*, utilizando a interface ELM obteve-se uma taxa de recepção de 0.69 Mbps, ao passo que, utilizando a interface TMAC, nas mesmas condições obteve-se uma taxa de 22.4 Mbps, o que corresponde a um factor de 32.5 vezes maior. Em relação aos testes de fiabilidade e de acessibilidade chegou-se à conclusão que ambas as interfaces permitem uma comunicação fiável e apresentam uma boa acessibilidade a partir do computador de teste. Tomando o desempenho como critério, recomendamos a implementação da interface TMAC em detrimento da interface ELM, embora aquela utilize mais recursos na FPGA. Foi também possível e controlar a placa ML605 a partir do computador de teste via *Ethernet*, utilizando a interface TMAC, com recurso a um servidor *Web* executado pelo sistema

embebido. Este teste mostrou, a interface TMAC é completamente funcional e que permite o controlo da placa a partir de um computador, situação similar à implementada no MobiDICK 4.

Os resultados obtidos no teste do módulo CRC implementado no testador MobiDICK 4, permitiram concluir que primeira versão testada funciona de forma parcialmente correcta. Tal como foi descrito no capítulo 5, a electrónica de frontaria realiza dois cálculos de CRCs, um valor de *CRC Global* e valores de CRC de cada dispositivo DMU. O testador MobiDICK 4 deverá ser capaz de realizar estes mesmos cálculos de forma a verificar a integridade de dados enviados pela electrónica de frontaria. Os primeiros testes permitiram-nos concluir que o módulo CRC é capaz de calcular correctamente o valor de *CRC Global* e realizar contagens de erros associados a ele. Os resultados obtidos para o *Super-drawer* de teste indicam uma percentagem de eventos com erros na gama de 13 a 15 %. Nos testes dos valores de CRC associados aos dispositivos DMU não se obtiveram os valores de CRC expectáveis. Uma análise mais detalhada permitiu verificar que o módulo CRC estava a funcionar correctamente e que os resultados obtidos deviam a uma incorrecta ordenação dos *bits* dos dados. A funcionalidade do módulo CRC tinha sido verificada em simulações antes e após a sua integração no ModiDICK 4. Recomenda-se novos testes e a sua adaptação a situações reais de aquisição de dados da electrónica de frontaria.

O trabalho ficará concluído com o teste da nova versão (já desenvolvida) que inclui a correcção ao problema observado no cálculo dos CRCs dos DMUs. Tal como já foi referido, novos testes serão efectuados, assim que o testador esteja disponível, para avaliar a correcção do problema, e assim o módulo CRC ficará completamente funcional no testador MobiDICK 4.

Apêndice A Tecnologia de Comunicação *Ethernet*

A.1 Modelo de Referência *Open Systems Interconnection* (OSI)

O modelo OSI para sistemas de comunicação foi estabelecido pela *International Standards Organization* (ISO):

Nível 7	Aplicação
Nível 6	Apresentação
Nível 5	Sessão
Nível 4	Transporte
Nível 3	Rede
Nível 2	Ligação de Dados
Nível 1	Físico

Tabela A.1: *Modelo de referência OSI.*

A tabela A.1 mostra os 7 níveis do modelo de referência OSI. A tecnologia *Ethernet* é implementada para fornecer serviços ao nível Físico e ao nível de Ligação de Dados.

Ligação de Dados

Para a tecnologia de comunicação *Ethernet* este nível é dividido em dois subníveis pela norma IEEE 802: o subnível de *Controlo Lógico de Ligação (Logical Link Control – LLC)* e o subnível de *Controlo de Acesso ao Meio (Medium Access Control – MAC)*.

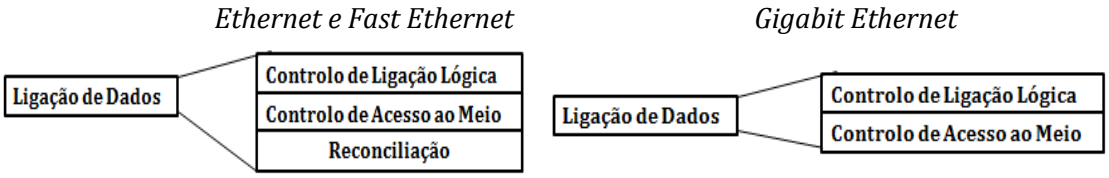


Figura A.1: *Subdivisão do nível de Ligação de Dados para a tecnologia Ethernet.*

Controlo de Ligação Lógica

Este subnível é definido no padrão IEEE 802.2 com o objectivo de permitir que o método de controlo da ligação seja independente de um método de acesso específico. As funções realizadas incluem a geração e a interpretação de comandos de controlo de fluxo e operações de recuperação quando um erro de transmissão é detectado [9].

Controlo de Acesso ao Meio (MAC)

Este subnível é responsável pela implementação do método de acesso ao meio físico. As principais funções incluem a verificação do canal de transmissão com o objectivo de monitorizar a sua ocupação, verificar a ocorrência de colisões, que ocorrem quando dois dispositivos transmitem dados simultaneamente através do canal em modo *half-duplex*, e executar algumas tarefas predefinidas caso uma colisão seja detectada.

Nível Físico

O nível físico apresenta quatro subníveis de acordo com as especificações do padrão IEEE 802, para sistemas *Ethernet* de 10 Mbps e 100 Mbps. Para os sistemas *Gigabit Ethernet* existe uma ligeira modificação.

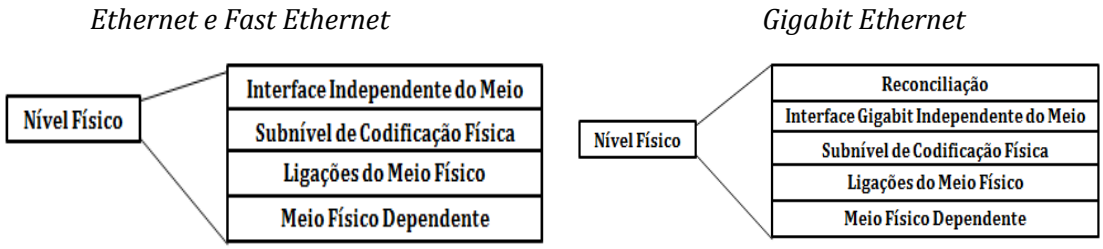


Figura A.2: *Subdivisão do nível de Físico para a tecnologia Ethernet.*

Interface Independente do Meio

Esta interface, que é normalmente chamada por MII *(Media Independent Interface).* Tem como função assegurar que qualquer tipo de meio físico padrão possa se comunicar de forma única, com o nível imediatamente a seguir, o nível de Ligação de Dados (subnível MAC).

Subnível de Codificação Física

Este subnível é mais conhecido por PCS *(Physical Coding Sublayer)*, apresenta como função assegurar a codificação e descodificação de dados para o nível de Controlo de Acesso ao Meio (MAC) e também dos dados enviados pelo nível MAC. Realiza as funções de transmissão e recepção da camada física, realiza também o processo de verificação do canal de transmissão *(Carrier Sense)* que será descrito posteriormente.

Ligações do Meio Físico

Este subnível é também conhecido por PMA *(Physical Coding Attachements)*, e representa a interface de comunicação com o meio físico. Apresenta como principal função serializar dados enviados do Subnível de Codificação Física num conjunto de *bits* que seja adequado ao meio

físico de transmissão. Apresenta também a capacidade de receber conjuntos de *bits* serializados da camada física e de os enviar ao Subnível de Codificação Física.

Meio Físico Dependente

Este subnível é normalmente conhecido por PMD *(Physical Medium Dependent)* e, tal como o nome indica é um subnível dependente do meio físico. É responsável pelos detalhes de transmissão e recepção de cada bit no meio físico. Apresenta como função a temporização dos *bits (bit timing)* e a codificação do sinal, sendo responsável pela interacção com o meio de comunicação (cabos coaxiais, fibra óptica, etc.). O subnível PMD liga-se ao meio de comunicação, por exemplo ao cabo coaxial, através de uma interface denominada de *Interface Dependente do Meio*, normalmente conhecida por MDI *(Medium Dependente Interface).* A interface MDI especifica os diversos tipos de conectores para diferentes tipos de meios físicos e dispositivos.

A.2 Componentes Principais

Há quatro componentes básicos que constituem um sistema de comunicação *Ethernet.* Esses quatro elementos são: o pacote de dados usualmente designado por *frame*, o protocolo de controlo de acesso ao meio, os dispositivos físicos de sinalização e o meio físico de transmissão de sinal [30].

A.2.1 Pacote de Dados Ethernet

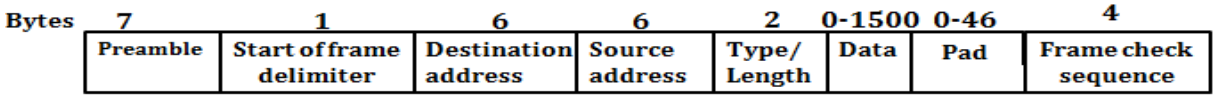


Figura A.3: *Pacote de dados Ethernet.*

Campo *Preamble*

Este campo usa 7 bytes, em que os *bits* são organizados de forma alternada entre 1-0 (isto é, 1010…). A função deste campo é anunciar a chegada de um pacote *Ethernet*, permitindo que todos os dispositivos da rede se sincronizem ao pacote antes que os dados importantes no endereço e no campo *Data* cheguem.

Campo *Start of Frame Delimiter (SDF)*

Este campo apresenta 1 byte de dimensão e é uma continuação do *Preamble*. A estrutura alternada de *bits* mantem-se, excepto para os dois últimos *bits*. Este campo deve conter o padrão de bits, 10101011. A quebra do padrão alternado nos dois últimos *bits* destrói o padrão de

sincronização do campo *Preambule* alertando a interface de que se seguem os endereços do emissor e do receptor e um pacote de dados no campo *Data*. Quando o controlador ou a interface *Ethernet* recebem um pacote, os campos *Preamble* e *SDF* são removidos, e os dados são guardados em memória. Da mesma forma, quando um controlador *Ethernet* transmite um pacote, insere esses dois campos no pacote antes de ele ser enviado.

Campo *Destination Address*

Este campo, como o próprio nome indica, corresponde ao endereço físico da interface *Ethernet* para o qual o pacote é enviado. Cada interface *Ethernet* apresenta um endereço único de 48 *bits* denominado de endereço físico ou endereço *MAC*, e usualmente apenas designado de *MAC Address* da interface.

Campo *Source Address*

O endereço local corresponde ao endereço físico da interface *Ethernet* que transmite o pacote de dados. Apresenta a mesma estrutura do endereço de destino, sendo que a dimensão é também de 48 *bits*. É um endereço único atribuído na altura de fabricação da própria interface *Ethernet* [30, 31].

Campo *Type/Length*

É um campo que apresenta 2 bytes de dimensão. Pode conter dois tipos de informação: o tipo de protocolo associado aos dados ou a dimensão em bytes dos dados do campo *Data*. O campo *Type* é utilizado no protocolo *Ethernet* original, ao passo que, o campo *Length* é utilizado no padrão IEEE 802.3. O campo *Type* indica qual é o protocolo associado aos dados inseridos no campo de dados *(Data)* que se segue: por exemplo um dos mais utilizados é o protocolo TCP/IP. A dimensão indica a dimensão em bytes dos dados que se encontram no campo *Data*.

Campo *Data*

Este campo contém os dados, o seu tamanho varia de 0 a 1500 bytes. Um pacote de dados *Ethernet* deve apresentar uma dimensão mínima de 64 bytes, devido a isso a dimensão mínima deste campo deve ser 46 bytes. Em muitas situações é incluído o campo *Pad* com um tamanho que varia entre 0 a 46 bytes, para garantir que o pacote tenha um comprimento mínimo de 64 bytes [30, 31].

Campo *Pad*

É utilizado para garantir que o pacote de dados *Ethernet* tenha no mínimo 64 bytes de tamanho, como foi referido anteriormente. Se o campo *Data* conter 0 bytes, o campo *Pad* terá 46 bytes. Se o campo de *Data* tiver 46 bytes ou mais, o campo *Pad* terá 0 bytes [30, 31].

Campo *Frame Check Sequence (FCS)*

Este campo contém um parâmetro que resulta da implementação de um mecanismo de detecção de erro, baseado no algoritmo *Cyclic Redundancy Check* (CRC) que entra em consideração o campo *Type/Length* e o campo *Data*. A interface *Ethernet* coloca neste campo o valor de CRC obtido. É utilizado o algoritmo CRC32.

A.2.2 Protocolo de Controlo de Acesso ao Meio

O protocolo de controlo de acesso ao meio *(MAC Protocol)* é um conjunto de regras que permitem que um conjunto de dispositivos ligados numa rede partilhe, de forma justa, um mesmo canal de comunicação *Ethernet*. O método de acesso ao meio primeiramente utilizado em redes *Ethernet (half-duplex)* foi o método *Carrier Sense Multiple Access/collision Detection* (CSMA/CD) [31].

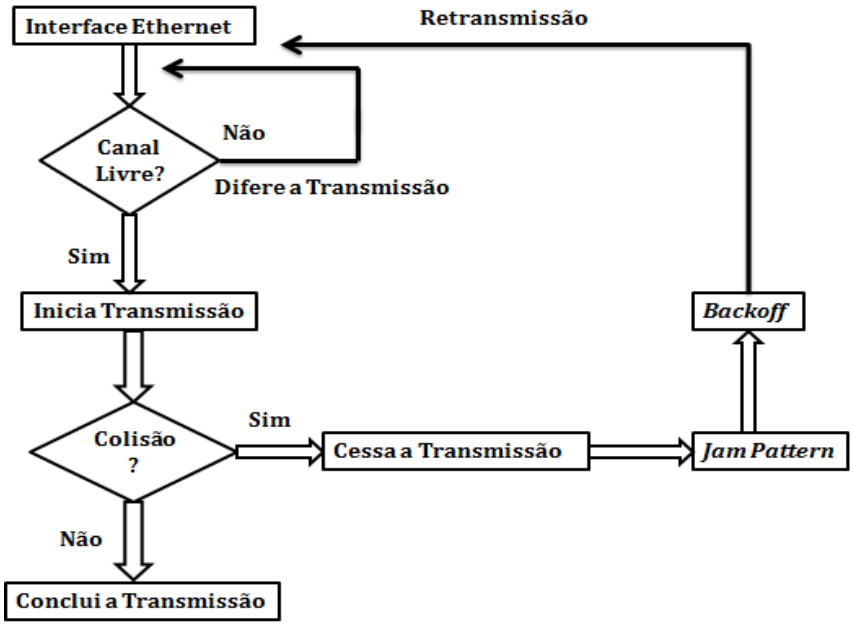


Figura A.4: *Método de Acesso CSMA/CD.*

No modelo de referência OSI, a camada de Ligação de Dados é dividida em duas subcamadas, a subcamada de *Controlo de Ligação Lógica (Logical Link Control)* e a subcamada de *Controlo de Acesso ao Meio (Médium Access Control-MAC)*. A subcamada MAC é responsável pela

verificação do canal, se este está disponível ou não, por verificar a ocorrência de colisões e realizar os processos apresentados na figura A.4. A tabela seguinte sumariza as principais funcionalidades de um controlador de acesso ao meio (MAC) que é implementado numa interface *Ethernet* [31].

Operações de Transmissão de Dados	➢ Aceitar dados da subcamada de Controlo de Ligação Lógica e construir o pacote *Ethernet*. ➢ Calcula o CRC e insere-o no campo FCS.
Controlo de Acesso ao Meio na Transmissão	➢ Deferir a transmissão se o canal estiver ocupado. ➢ Transmitir depois de passar o tempo de *Interframe Gap.* ➢ Apresentar o pacote *Ethernet* à camada física para a transmissão. ➢ Parar a transmissão se o meio estiver ocupado. ➢ Transmitir o sinal *jam pattern* para transmitir a informação sobre a ocorrência de uma colisão. ➢ Reprogramar retransmissões depois de uma colisão até ter sucesso ou até atingir o limite definido.
Operações de Recepção de Dados	➢ Descartar todos os pacotes que não são endereçados para a interface de recepção. ➢ Reconhecer todos os pacotes *broadcast* e os pacotes enviados especificamente para a interface. ➢ Calcular o valor de CRC. ➢ Remover todos os campos do pacote de dados excepto o campo de dados. ➢ Passar os dados para a subcamada de Controlo de Ligação lógica (LLC).
Controlo de Acesso ao Meio na Recepção	➢ Receber um conjunto de *bits* da camada física. ➢ Verificar a dimensão do pacote. ➢ Descartar os pacotes que não tenham uma dimensão em bytes par ou que têm uma dimensão menor do que a dimensão mínima de um pacote.

Tabela A.2: *Áreas funcionais do MAC.*

A comunicação em modo *full-duplex* é implementada apenas com dois dispositivos nos terminais de um canal, no qual se pode transmitir e receber dados simultaneamente. A tecnologia de comunicação *Ethernet* foi originalmente implementada como um método de transmissão em redes locais em modo *half-duplex*. No modo *full-duplex* a interface *Ethernet* não utiliza o método de acesso CSMA/CD, pois a rede é constituída por apenas dois dispositivos que podem transmitir dados simultaneamente através do mesmo canal. O referido método de controlo de acesso ao meio é apenas utilizado em redes locais em modo *half-duplex*. Para o modo

de funcionamento em *full-duplex*, a norma IEEE 802.3x introduz também um mecanismo de controlo, em tempo real, de transmissão e recepção de pacotes numa interface *Ethernet* [2]. Esse mecanismo é designado por Protocolo de Controlo do MAC *(MAC Control Protocol)*. Este protocolo fornece um mecanismo no qual as interfaces *Ethernet* recebem um pacote específico para o controlo de fluxo de dados, denominado de pacote de controlo de fluxo. O Protocolo de Controlo do MAC é implementado de forma a permitir uma interacção em tempo real para controlo do fluxo de dados [30]. O processo de controlo de fluxo é utilizado para evitar o congestionamento da rede no modo *full-duplex*, sendo também utilizado quando uma interface já não tem capacidade de armazenamento de dados nos seus registos internos. Uma interface que receber uma notificação de congestionamento de rede, suspende a transmissão durante um certo intervalo de tempo. Para o modo *full-duplex* é implementado um mecanismo chamado autonegociação. A autonegociação é um mecanismo através do qual duas interfaces *Ethernet* ligadas em modo *full-duplex* negociam entre si e escolhem parâmetros comuns de comunicação. As duas interfaces escolhem as melhores opções suportadas por ambas, tais opções incluem: maior velocidade de ligação possível, o modo *duplex* e a opção de controlo de fluxo.

A.2.3 Componentes de *Hardware* do Sistema Ethernet

A tecnologia *Ethernet* original é implementada utilizando cinco componentes de *hardware*: Controlador *Ethernet*, conhecido também como *Network Interface Card* (NIC), *Transceiver* e o respectivo cabo, o Cabo Coaxial como meio de transmissão e um dispositivo chamado de *Tap* que serve de conexão entre o *Transceiver* e o meio de transmissão.

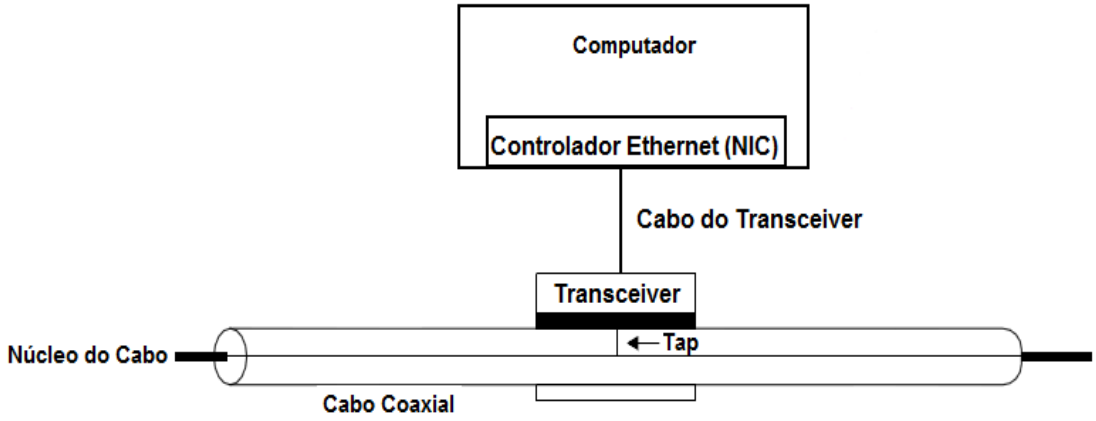


Figura A.5: *Componentes de hardware de um sistema Ethernet.*

De seguida é apresentada as funcionalidades de cada um desses componentes [31]:

Controlador *Ethernet* (NIC)

O Controlador *Ethernet* também conhecido como *Network Interface Card* (NIC) representa uma placa electrónica implementada no dispositivo de comunicação (inicialmente um computador). É responsável pela formatação de dados em pacotes de dados *Ethernet (frames)* e pelo cálculo de CRC para a detecção de erro. É responsável pela transmissão de pacotes de dados para o *Transceiver* e pela recepção de dados enviados pelo *Transceiver*.

Cabo Coaxial

O cabo coaxial foi a selecção original para ser o meio de transmissão da tecnologia *Ethernet*. A tecnologia *Ethernet* foi desenvolvida inicialmente para interconectar computadores em diferentes locais, o cabo coaxial seria a melhor opção para satisfazer esse requisito, pois permite conectar dispositivos de comunicação a grandes distâncias.

Transceiver e Cabo do *Transceiver*

A palavra *Transceiver* é uma combinação das palavras *Transmitter* e *Receiver*. Um *transceiver* contém a electrónica necessária para receber e transmitir os sinais através do meio transmissão, o cabo coaxial. Os dispositivos *Transceiver* incluem um dispositivo removível normalmente chamado *Tap*. Os dispositivos *Tap* permitem o contacto como o núcleo do cabo coaxial. O conjunto *Transceiver + Tap*, é também conhecida tecnicamente como *Medium Attachment Unit* (Unidade de Ligação ao Meio). O dispositivo *Transceiver* é também responsável pelo processo de *Carrier Sense* e de detecção de colisão. Quando detecta uma colisão, o *Transceiver* é responsável pela transmissão do sinal *Jam,* descrito anteriormente. O cabo do *Transceiver* permite conectar o Controlador *Ethernet* ao *Transceiver*.

Apêndice B Resultados obtidos utilizando a aplicação Iperf e o Teste PING

B.1 Taxas de Transmissão e Recepção de Dados

B.1.1 Interface TMAC

Raw API

a) Capacidade de Ligação: 1 Gbps

Transmissão

```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 64.0 KByte (default)
------------------------------------------------------------
[  4] local 192.168.1.100 port 5001 connected with 192.168.1.10 port 4097
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0- 5.0 sec  63.3 MBytes   106 Mbits/sec
[  4]  5.0-10.0 sec  63.6 MBytes   107 Mbits/sec
[  4] 10.0-15.0 sec  63.8 MBytes   107 Mbits/sec
[  4] 15.0-20.0 sec  63.7 MBytes   107 Mbits/sec
[  4] 20.0-25.0 sec  63.1 MBytes   106 Mbits/sec
[  4] 25.0-30.0 sec  63.6 MBytes   107 Mbits/sec
[  4] 30.0-35.0 sec  63.4 MBytes   106 Mbits/sec
[  4] 35.0-40.0 sec  63.7 MBytes   107 Mbits/sec
[  4] 40.0-45.0 sec  63.8 MBytes   107 Mbits/sec
[  4] 45.0-50.0 sec  64.3 MBytes   108 Mbits/sec
[  4] 50.0-55.0 sec  63.5 MBytes   107 Mbits/sec
[  4] 55.0-60.0 sec  64.1 MBytes   108 Mbits/sec
[  4] 60.0-65.0 sec  63.3 MBytes   106 Mbits/sec
[  4] 65.0-70.0 sec  65.2 MBytes   109 Mbits/sec
[  4] 70.0-75.0 sec  64.0 MBytes   107 Mbits/sec
[  4] 75.0-80.0 sec  63.8 MBytes   107 Mbits/sec
[  4] 80.0-85.0 sec  63.9 MBytes   107 Mbits/sec
```

Recepção

```
------------------------------------------------------------
Client connecting to 192.168.1.10, TCP port 5001
TCP window size: 8.00 KByte
------------------------------------------------------------
[  3] local 192.168.1.100 port 51114 connected with 192.168.1.10 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 5.0 sec  71.6 MBytes   120 Mbits/sec
[  3]  5.0-10.0 sec  71.8 MBytes   120 Mbits/sec
[  3] 10.0-15.0 sec  72.0 MBytes   121 Mbits/sec
[  3] 15.0-20.0 sec  72.0 MBytes   121 Mbits/sec
[  3] 20.0-25.0 sec  71.9 MBytes   121 Mbits/sec
[  3] 25.0-30.0 sec  72.0 MBytes   121 Mbits/sec
[  3] 30.0-35.0 sec  71.2 MBytes   120 Mbits/sec
[  3] 35.0-40.0 sec  72.2 MBytes   121 Mbits/sec
[  3] 40.0-45.0 sec  72.0 MBytes   121 Mbits/sec
[  3] 45.0-50.0 sec  71.6 MBytes   120 Mbits/sec
[  3]  0.0-50.0 sec   718 MBytes   121 Mbits/sec
```

b) Capacidade de ligação: 100 Mbps

Transmissão

```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 64.0 KByte (default)
------------------------------------------------------------
[  4] local 192.168.1.100 port 5001 connected with 192.168.1.10 port 4097
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0- 5.0 sec  33.9 MBytes  56.8 Mbits/sec
[  4]  5.0-10.0 sec  33.0 MBytes  55.4 Mbits/sec
[  4] 10.0-15.0 sec  33.3 MBytes  55.9 Mbits/sec
[  4] 15.0-20.0 sec  33.9 MBytes  56.9 Mbits/sec
[  4] 20.0-25.0 sec  33.0 MBytes  55.4 Mbits/sec
[  4] 25.0-30.0 sec  33.2 MBytes  55.7 Mbits/sec
[  4] 30.0-35.0 sec  34.0 MBytes  57.1 Mbits/sec
[  4] 35.0-40.0 sec  33.1 MBytes  55.6 Mbits/sec
[  4] 40.0-45.0 sec  33.5 MBytes  56.2 Mbits/sec
[  4] 45.0-50.0 sec  33.9 MBytes  56.8 Mbits/sec
[  4] 50.0-55.0 sec  33.3 MBytes  55.9 Mbits/sec
[  4] 55.0-60.0 sec  33.7 MBytes  56.5 Mbits/sec
[  4] 60.0-65.0 sec  33.5 MBytes  56.3 Mbits/sec
[  4] 65.0-70.0 sec  33.3 MBytes  55.9 Mbits/sec
[  4] 70.0-75.0 sec  33.9 MBytes  56.8 Mbits/sec
[  4] 75.0-80.0 sec  33.5 MBytes  56.3 Mbits/sec
```

Recepção

```
------------------------------------------------------------
Client connecting to 192.168.1.10, TCP port 5001
TCP window size: 8.00 KByte
------------------------------------------------------------
[  3] local 192.168.1.100 port 51469 connected with 192.168.1.10 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 5.0 sec  37.8 MBytes  63.3 Mbits/sec
[  3]  5.0-10.0 sec  32.4 MBytes  54.3 Mbits/sec
[  3] 10.0-15.0 sec  32.8 MBytes  54.9 Mbits/sec
[  3] 15.0-20.0 sec  32.1 MBytes  53.9 Mbits/sec
[  3] 20.0-25.0 sec  32.2 MBytes  54.1 Mbits/sec
[  3] 25.0-30.0 sec  32.5 MBytes  54.5 Mbits/sec
[  3] 30.0-35.0 sec  32.2 MBytes  54.1 Mbits/sec
[  3] 35.0-40.0 sec  32.5 MBytes  54.5 Mbits/sec
[  3] 40.0-45.0 sec  32.5 MBytes  54.5 Mbits/sec
[  3] 45.0-50.0 sec  32.2 MBytes  54.1 Mbits/sec
[  3]  0.0-50.0 sec   329 MBytes  55.2 Mbits/sec
```

c) Capacidade de ligação: 10 Mbps

Transmissão

```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 64.0 KByte (default)
------------------------------------------------------------
[  4] local 192.168.1.100 port 5001 connected with 192.168.1.10 port 4097
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0- 5.0 sec  5.64 MBytes  9.46 Mbits/sec
[  4]  5.0-10.0 sec  5.63 MBytes  9.45 Mbits/sec
[  4] 10.0-15.0 sec  5.63 MBytes  9.44 Mbits/sec
[  4] 15.0-20.0 sec  5.59 MBytes  9.38 Mbits/sec
[  4] 20.0-25.0 sec  5.61 MBytes  9.42 Mbits/sec
[  4] 25.0-30.0 sec  5.62 MBytes  9.44 Mbits/sec
[  4] 30.0-35.0 sec  5.61 MBytes  9.41 Mbits/sec
[  4] 35.0-40.0 sec  5.59 MBytes  9.38 Mbits/sec
[  4] 40.0-45.0 sec  5.63 MBytes  9.45 Mbits/sec
[  4] 45.0-50.0 sec  5.62 MBytes  9.44 Mbits/sec
[  4] 50.0-55.0 sec  5.61 MBytes  9.41 Mbits/sec
[  4] 55.0-60.0 sec  5.61 MBytes  9.42 Mbits/sec
[  4] 60.0-65.0 sec  5.64 MBytes  9.46 Mbits/sec
[  4] 65.0-70.0 sec  5.63 MBytes  9.44 Mbits/sec
[  4] 70.0-75.0 sec  5.63 MBytes  9.44 Mbits/sec
```

Recepção

```
------------------------------------------------------------
Client connecting to 192.168.1.10, TCP port 5001
TCP window size: 8.00 KByte
------------------------------------------------------------
[  3] local 192.168.1.100 port 51479 connected with 192.168.1.10 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 5.0 sec  5.62 MBytes  9.44 Mbits/sec
[  3]  5.0-10.0 sec  5.62 MBytes  9.44 Mbits/sec
[  3] 10.0-15.0 sec  5.62 MBytes  9.44 Mbits/sec
[  3] 15.0-20.0 sec  5.62 MBytes  9.44 Mbits/sec
[  3] 20.0-25.0 sec  5.50 MBytes  9.23 Mbits/sec
[  3] 25.0-30.0 sec  5.75 MBytes  9.65 Mbits/sec
[  3] 30.0-35.0 sec  5.62 MBytes  9.44 Mbits/sec
[  3] 35.0-40.0 sec  5.62 MBytes  9.44 Mbits/sec
[  3] 40.0-45.0 sec  5.62 MBytes  9.44 Mbits/sec
[  3] 45.0-50.0 sec  5.62 MBytes  9.44 Mbits/sec
[  3]  0.0-50.1 sec  56.4 MBytes  9.44 Mbits/sec
```

Socket API

a) Capacidade de ligação: 1 Gbps

Transmissão

```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 64.0 KByte (default)
------------------------------------------------------------
[  4] local 192.168.1.100 port 5001 connected with 192.168.1.10 port 4097
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0- 5.0 sec  22.2 MBytes  37.2 Mbits/sec
[  4]  5.0-10.0 sec  22.1 MBytes  37.1 Mbits/sec
[  4] 10.0-15.0 sec  22.2 MBytes  37.3 Mbits/sec
[  4] 15.0-20.0 sec  22.3 MBytes  37.4 Mbits/sec
[  4] 20.0-25.0 sec  22.2 MBytes  37.2 Mbits/sec
[  4] 25.0-30.0 sec  22.2 MBytes  37.3 Mbits/sec
[  4] 30.0-35.0 sec  22.3 MBytes  37.5 Mbits/sec
[  4] 35.0-40.0 sec  22.1 MBytes  37.2 Mbits/sec
[  4] 40.0-45.0 sec  22.2 MBytes  37.2 Mbits/sec
[  4] 45.0-50.0 sec  22.2 MBytes  37.2 Mbits/sec
[  4] 50.0-55.0 sec  22.1 MBytes  37.1 Mbits/sec
[  4] 55.0-60.0 sec  22.1 MBytes  37.1 Mbits/sec
[  4] 60.0-65.0 sec  22.2 MBytes  37.3 Mbits/sec
[  4] 65.0-70.0 sec  22.4 MBytes  37.5 Mbits/sec
[  4] 70.0-75.0 sec  22.3 MBytes  37.5 Mbits/sec
[  4] 75.0-80.0 sec  22.4 MBytes  37.6 Mbits/sec
[  4] 80.0-85.0 sec  22.4 MBytes  37.6 Mbits/sec
```

Recepção

```
Client connecting to 192.168.1.10, TCP port 5001
TCP window size: 8.00 KByte
------------------------------------------------------------
[  3] local 192.168.1.100 port 51487 connected with 192.168.1.10 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 5.0 sec  13.0 MBytes  21.8 Mbits/sec
[  3]  5.0-10.0 sec  13.1 MBytes  22.0 Mbits/sec
[  3] 10.0-15.0 sec  13.1 MBytes  22.0 Mbits/sec
[  3] 15.0-20.0 sec  13.1 MBytes  22.0 Mbits/sec
[  3] 20.0-25.0 sec  13.0 MBytes  21.8 Mbits/sec
[  3] 25.0-30.0 sec  13.1 MBytes  22.0 Mbits/sec
[  3] 30.0-35.0 sec  13.1 MBytes  22.0 Mbits/sec
[  3] 35.0-40.0 sec  13.0 MBytes  21.8 Mbits/sec
[  3] 40.0-45.0 sec  13.1 MBytes  22.0 Mbits/sec
[  3] 45.0-50.0 sec  13.1 MBytes  22.0 Mbits/sec
[  3]  0.0-50.0 sec   131 MBytes  22.0 Mbits/sec
```

b) Capacidade de ligação: 100 Mbps

Transmissão

```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 64.0 KByte (default)
------------------------------------------------------------
[  4] local 192.168.1.100 port 5001 connected with 192.168.1.10 port 4097
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0- 5.0 sec  22.9 MBytes  38.3 Mbits/sec
[  4]  5.0-10.0 sec  22.8 MBytes  38.3 Mbits/sec
[  4] 10.0-15.0 sec  22.8 MBytes  38.3 Mbits/sec
[  4] 15.0-20.0 sec  22.8 MBytes  38.2 Mbits/sec
[  4] 20.0-25.0 sec  22.8 MBytes  38.2 Mbits/sec
[  4] 25.0-30.0 sec  22.9 MBytes  38.3 Mbits/sec
[  4] 30.0-35.0 sec  22.9 MBytes  38.3 Mbits/sec
[  4] 35.0-40.0 sec  22.8 MBytes  38.2 Mbits/sec
[  4] 40.0-45.0 sec  22.8 MBytes  38.3 Mbits/sec
[  4] 45.0-50.0 sec  22.7 MBytes  38.0 Mbits/sec
[  4] 50.0-55.0 sec  22.8 MBytes  38.3 Mbits/sec
[  4] 55.0-60.0 sec  22.8 MBytes  38.2 Mbits/sec
[  4] 60.0-65.0 sec  22.7 MBytes  38.1 Mbits/sec
[  4] 65.0-70.0 sec  22.9 MBytes  38.4 Mbits/sec
[  4] 70.0-75.0 sec  23.0 MBytes  38.6 Mbits/sec
[  4] 75.0-80.0 sec  23.1 MBytes  38.7 Mbits/sec
```

Recepção

```
------------------------------------------------------------
Client connecting to 192.168.1.10, TCP port 5001
TCP window size: 8.00 KByte
------------------------------------------------------------
[  3] local 192.168.1.100 port 51494 connected with 192.168.1.10 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 5.0 sec  13.4 MBytes  22.4 Mbits/sec
[  3]  5.0-10.0 sec  13.2 MBytes  22.2 Mbits/sec
[  3] 10.0-15.0 sec  13.4 MBytes  22.4 Mbits/sec
[  3] 15.0-20.0 sec  13.2 MBytes  22.2 Mbits/sec
[  3] 20.0-25.0 sec  13.4 MBytes  22.4 Mbits/sec
[  3] 25.0-30.0 sec  13.4 MBytes  22.4 Mbits/sec
[  3] 30.0-35.0 sec  13.2 MBytes  22.2 Mbits/sec
[  3] 35.0-40.0 sec  13.2 MBytes  22.2 Mbits/sec
[  3] 40.0-45.0 sec  13.4 MBytes  22.4 Mbits/sec
[  3] 45.0-50.0 sec  13.2 MBytes  22.2 Mbits/sec
[  3]  0.0-50.0 sec   133 MBytes  22.4 Mbits/sec
```

c) Capacidade de ligação: 10 Mbps

Transmissão

```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 64.0 KByte (default)
------------------------------------------------------------
[  4] local 192.168.1.100 port 5001 connected with 192.168.1.10 port 4097
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0- 5.0 sec  5.62 MBytes  9.44 Mbits/sec
[  4]  5.0-10.0 sec  5.64 MBytes  9.46 Mbits/sec
[  4] 10.0-15.0 sec  5.62 MBytes  9.43 Mbits/sec
[  4] 15.0-20.0 sec  5.62 MBytes  9.43 Mbits/sec
[  4] 20.0-25.0 sec  5.64 MBytes  9.46 Mbits/sec
[  4] 25.0-30.0 sec  5.61 MBytes  9.41 Mbits/sec
[  4] 30.0-35.0 sec  5.63 MBytes  9.45 Mbits/sec
[  4] 35.0-40.0 sec  5.64 MBytes  9.46 Mbits/sec
[  4] 40.0-45.0 sec  5.63 MBytes  9.44 Mbits/sec
[  4] 45.0-50.0 sec  5.62 MBytes  9.43 Mbits/sec
[  4] 50.0-55.0 sec  5.63 MBytes  9.45 Mbits/sec
[  4] 55.0-60.0 sec  5.62 MBytes  9.44 Mbits/sec
[  4] 60.0-65.0 sec  5.62 MBytes  9.43 Mbits/sec
[  4] 65.0-70.0 sec  5.66 MBytes  9.49 Mbits/sec
[  4] 70.0-75.0 sec  5.64 MBytes  9.46 Mbits/sec
[  4] 75.0-80.0 sec  5.65 MBytes  9.47 Mbits/sec
[  4] 80.0-85.0 sec  5.65 MBytes  9.48 Mbits/sec
```

Recepção

```
------------------------------------------------------------
Client connecting to 192.168.1.10, TCP port 5001
TCP window size: 8.00 KByte
------------------------------------------------------------
[  3] local 192.168.1.100 port 51397 connected with 192.168.1.10 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 5.0 sec  5.62 MBytes  9.44 Mbits/sec
[  3]  5.0-10.0 sec  5.50 MBytes  9.23 Mbits/sec
[  3] 10.0-15.0 sec  5.62 MBytes  9.44 Mbits/sec
[  3] 15.0-20.0 sec  5.62 MBytes  9.44 Mbits/sec
[  3] 20.0-25.0 sec  5.62 MBytes  9.44 Mbits/sec
[  3] 25.0-30.0 sec  5.62 MBytes  9.44 Mbits/sec
[  3] 30.0-35.0 sec  5.62 MBytes  9.44 Mbits/sec
[  3] 35.0-40.0 sec  5.62 MBytes  9.44 Mbits/sec
[  3] 40.0-45.0 sec  5.62 MBytes  9.44 Mbits/sec
[  3] 45.0-50.0 sec  5.62 MBytes  9.44 Mbits/sec
[  3]  0.0-50.0 sec  56.2 MBytes  9.43 Mbits/sec
```

Teste PING

```
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255

Ping statistics for 192.168.1.10:
    Packets: Sent = 100, Received = 100, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

B.1.2 Interface ELM

Socket API

a) Capacidade de ligação: 10 Mbps

Transmissão

```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 64.0 KByte (default)
------------------------------------------------------------
[  4] local 192.168.1.100 port 5001 connected with 192.168.1.10 port 4097
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0- 5.0 sec  1.96 MBytes  3.29 Mbits/sec
[  4]  5.0-10.0 sec  1.97 MBytes  3.30 Mbits/sec
[  4] 10.0-15.0 sec  1.98 MBytes  3.32 Mbits/sec
[  4] 15.0-20.0 sec  1.97 MBytes  3.31 Mbits/sec
[  4] 20.0-25.0 sec  1.96 MBytes  3.28 Mbits/sec
[  4] 25.0-30.0 sec  1.97 MBytes  3.30 Mbits/sec
[  4] 30.0-35.0 sec  1.97 MBytes  3.30 Mbits/sec
[  4] 35.0-40.0 sec  1.97 MBytes  3.30 Mbits/sec
[  4] 40.0-45.0 sec  1.97 MBytes  3.31 Mbits/sec
[  4] 45.0-50.0 sec  1.97 MBytes  3.31 Mbits/sec
[  4] 50.0-55.0 sec  1.97 MBytes  3.30 Mbits/sec
[  4] 55.0-60.0 sec  1.96 MBytes  3.29 Mbits/sec
[  4] 60.0-65.0 sec  1.97 MBytes  3.31 Mbits/sec
[  4] 65.0-70.0 sec  1.98 MBytes  3.32 Mbits/sec
[  4] 70.0-75.0 sec  1.98 MBytes  3.32 Mbits/sec
[  4] 75.0-80.0 sec  1.97 MBytes  3.30 Mbits/sec
[  4] 80.0-85.0 sec  1.97 MBytes  3.31 Mbits/sec
[  4] 85.0-90.0 sec  1.97 MBytes  3.30 Mbits/sec
[  4] 90.0-95.0 sec  1.98 MBytes  3.32 Mbits/sec
```

Transmissão

```
------------------------------------------------------------
Client connecting to 192.168.1.10, TCP port 5001
TCP window size: 8.00 KByte
------------------------------------------------------------
[  3] local 192.168.1.100 port 51634 connected with 192.168.1.10 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 5.0 sec   256 KBytes   419 Kbits/sec
[  3]  5.0-10.0 sec   384 KBytes   629 Kbits/sec
[  3] 10.0-15.0 sec   384 KBytes   629 Kbits/sec
[  3] 15.0-20.0 sec   256 KBytes   419 Kbits/sec
[  3] 20.0-25.0 sec   384 KBytes   629 Kbits/sec
[  3] 25.0-30.0 sec   384 KBytes   629 Kbits/sec
[  3] 30.0-35.0 sec   384 KBytes   629 Kbits/sec
[  3] 35.0-40.0 sec   256 KBytes   419 Kbits/sec
[  3] 40.0-45.0 sec   384 KBytes   629 Kbits/sec
[  3] 45.0-50.0 sec   384 KBytes   629 Kbits/sec
[  3]  0.0-51.4 sec  3.50 MBytes   571 Kbits/sec
```

Teste PING

```
Reply from 192.168.1.10: bytes=32 time=18ms TTL=255
Reply from 192.168.1.10: bytes=32 time=20ms TTL=255
Reply from 192.168.1.10: bytes=32 time=11ms TTL=255
Reply from 192.168.1.10: bytes=32 time=22ms TTL=255
Reply from 192.168.1.10: bytes=32 time=23ms TTL=255
Reply from 192.168.1.10: bytes=32 time=23ms TTL=255
Reply from 192.168.1.10: bytes=32 time=4ms TTL=255
Reply from 192.168.1.10: bytes=32 time=13ms TTL=255
Reply from 192.168.1.10: bytes=32 time=11ms TTL=255
Reply from 192.168.1.10: bytes=32 time=11ms TTL=255
Reply from 192.168.1.10: bytes=32 time=12ms TTL=255
Reply from 192.168.1.10: bytes=32 time=13ms TTL=255
Reply from 192.168.1.10: bytes=32 time=14ms TTL=255
Reply from 192.168.1.10: bytes=32 time=15ms TTL=255
Reply from 192.168.1.10: bytes=32 time=17ms TTL=255
Reply from 192.168.1.10: bytes=32 time=18ms TTL=255
Reply from 192.168.1.10: bytes=32 time=19ms TTL=255
Reply from 192.168.1.10: bytes=32 time=20ms TTL=255

Ping statistics for 192.168.1.10:
    Packets: Sent = 100, Received = 100, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 26ms, Average = 13ms
```

b) Capacidade de ligação: 100 Mbps

Transmissão

```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 64.0 KByte (default)
------------------------------------------------------------
[  4] local 192.168.1.100 port 5001 connected with 192.168.1.10 port 4097
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0- 5.0 sec  9.95 MBytes  16.7 Mbits/sec
[  4]  5.0-10.0 sec  10.1 MBytes  17.0 Mbits/sec
[  4] 10.0-15.0 sec  10.0 MBytes  16.8 Mbits/sec
[  4] 15.0-20.0 sec  10.0 MBytes  16.8 Mbits/sec
[  4] 20.0-25.0 sec  10.1 MBytes  16.9 Mbits/sec
[  4] 25.0-30.0 sec  9.97 MBytes  16.7 Mbits/sec
[  4] 30.0-35.0 sec  10.0 MBytes  16.8 Mbits/sec
[  4] 35.0-40.0 sec  9.92 MBytes  16.6 Mbits/sec
[  4] 40.0-45.0 sec  10.0 MBytes  16.8 Mbits/sec
[  4] 45.0-50.0 sec  10.0 MBytes  16.8 Mbits/sec
[  4] 50.0-55.0 sec  10.1 MBytes  16.9 Mbits/sec
[  4] 55.0-60.0 sec  10.0 MBytes  16.8 Mbits/sec
[  4] 60.0-65.0 sec  10.2 MBytes  17.0 Mbits/sec
[  4] 65.0-70.0 sec  10.0 MBytes  16.8 Mbits/sec
[  4] 70.0-75.0 sec  10.1 MBytes  17.0 Mbits/sec
[  4] 75.0-80.0 sec  10.1 MBytes  17.0 Mbits/sec
[  4] 80.0-85.0 sec  9.96 MBytes  16.7 Mbits/sec
[  4] 85.0-90.0 sec  9.91 MBytes  16.6 Mbits/sec
```

Transmissão

```
------------------------------------------------------------
Client connecting to 192.168.1.10, TCP port 5001
TCP window size: 8.00 KByte
------------------------------------------------------------
[  3] local 192.168.1.100 port 50530 connected with 192.168.1.10 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 5.0 sec   384 KBytes   629 Kbits/sec
[  3]  5.0-10.0 sec   512 KBytes   839 Kbits/sec
[  3] 10.0-15.0 sec   384 KBytes   629 Kbits/sec
[  3] 15.0-20.0 sec   384 KBytes   629 Kbits/sec
[  3] 20.0-25.0 sec   512 KBytes   839 Kbits/sec
[  3] 25.0-30.0 sec   384 KBytes   629 Kbits/sec
[  3] 30.0-35.0 sec   512 KBytes   839 Kbits/sec
[  3] 35.0-40.0 sec   384 KBytes   629 Kbits/sec
[  3] 40.0-45.0 sec   512 KBytes   839 Kbits/sec
[  3] 45.0-50.0 sec   384 KBytes   629 Kbits/sec
[  3]  0.0-50.2 sec  4.38 MBytes   731 Kbits/sec
```

Teste PING

```
Reply from 192.168.1.10: bytes=32 time=6ms TTL=255
Reply from 192.168.1.10: bytes=32 time=6ms TTL=255
Reply from 192.168.1.10: bytes=32 time=6ms TTL=255
Reply from 192.168.1.10: bytes=32 time=6ms TTL=255
Reply from 192.168.1.10: bytes=32 time=7ms TTL=255
Reply from 192.168.1.10: bytes=32 time=7ms TTL=255
Reply from 192.168.1.10: bytes=32 time=7ms TTL=255
Reply from 192.168.1.10: bytes=32 time=7ms TTL=255
Reply from 192.168.1.10: bytes=32 time=9ms TTL=255
Reply from 192.168.1.10: bytes=32 time=9ms TTL=255
Reply from 192.168.1.10: bytes=32 time=10ms TTL=255
Reply from 192.168.1.10: bytes=32 time=2ms TTL=255
Reply from 192.168.1.10: bytes=32 time=1ms TTL=255
Reply from 192.168.1.10: bytes=32 time=2ms TTL=255
Reply from 192.168.1.10: bytes=32 time=4ms TTL=255
Reply from 192.168.1.10: bytes=32 time=5ms TTL=255
Reply from 192.168.1.10: bytes=32 time=6ms TTL=255
Reply from 192.168.1.10: bytes=32 time=7ms TTL=255

Ping statistics for 192.168.1.10:
    Packets: Sent = 100, Received = 100, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 10ms, Average = 5ms
```

RAW API

a) Capacidade de ligação: 10 Mbps

Transmissão

```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 64.0 KByte (default)
------------------------------------------------------------
[  4] local 192.168.1.100 port 5001 connected with 192.168.1.10 port 4097
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0- 5.0 sec  4.88 MBytes  8.19 Mbits/sec
[  4]  5.0-10.0 sec  5.11 MBytes  8.58 Mbits/sec
[  4] 10.0-15.0 sec  5.13 MBytes  8.61 Mbits/sec
[  4] 15.0-20.0 sec  5.12 MBytes  8.59 Mbits/sec
[  4] 20.0-25.0 sec  5.13 MBytes  8.61 Mbits/sec
[  4] 25.0-30.0 sec  5.12 MBytes  8.58 Mbits/sec
[  4] 30.0-35.0 sec  5.13 MBytes  8.61 Mbits/sec
[  4] 35.0-40.0 sec  5.12 MBytes  8.59 Mbits/sec
[  4] 40.0-45.0 sec  5.11 MBytes  8.58 Mbits/sec
[  4] 45.0-50.0 sec  5.12 MBytes  8.60 Mbits/sec
[  4] 50.0-55.0 sec  5.11 MBytes  8.57 Mbits/sec
[  4] 55.0-60.0 sec  5.11 MBytes  8.57 Mbits/sec
[  4] 60.0-65.0 sec  5.13 MBytes  8.60 Mbits/sec
[  4] 65.0-70.0 sec  5.13 MBytes  8.60 Mbits/sec
[  4] 70.0-75.0 sec  5.13 MBytes  8.61 Mbits/sec
[  4] 75.0-80.0 sec  5.12 MBytes  8.59 Mbits/sec
[  4] 80.0-85.0 sec  5.12 MBytes  8.59 Mbits/sec
```

Recepção

```
------------------------------------------------------------
Client connecting to 192.168.1.10, TCP port 5001
TCP window size: 8.00 KByte
------------------------------------------------------------
[  3] local 192.168.1.100 port 51760 connected with 192.168.1.10 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 5.0 sec  3.50 MBytes  5.87 Mbits/sec
[  3]  5.0-10.0 sec  1.88 MBytes  3.15 Mbits/sec
[  3] 10.0-15.0 sec  3.25 MBytes  5.45 Mbits/sec
[  3] 15.0-20.0 sec  3.62 MBytes  6.08 Mbits/sec
[  3] 20.0-25.0 sec  2.00 MBytes  3.36 Mbits/sec
[  3] 25.0-30.0 sec   768 KBytes  1.26 Mbits/sec
[  3] 30.0-35.0 sec  2.38 MBytes  3.98 Mbits/sec
[  3] 35.0-40.0 sec  3.12 MBytes  5.24 Mbits/sec
[  3] 40.0-45.0 sec  3.38 MBytes  5.66 Mbits/sec
[  3] 45.0-50.0 sec  2.12 MBytes  3.57 Mbits/sec
[  3]  0.0-50.2 sec  26.1 MBytes  4.37 Mbits/sec
```

Teste PING

```
Reply from 192.168.1.10: bytes=32 time=2ms TTL=255
Reply from 192.168.1.10: bytes=32 time=4ms TTL=255
Reply from 192.168.1.10: bytes=32 time=5ms TTL=255
Reply from 192.168.1.10: bytes=32 time=6ms TTL=255
Reply from 192.168.1.10: bytes=32 time=8ms TTL=255
Reply from 192.168.1.10: bytes=32 time=9ms TTL=255
Reply from 192.168.1.10: bytes=32 time=11ms TTL=255
Reply from 192.168.1.10: bytes=32 time=12ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time=3ms TTL=255
Reply from 192.168.1.10: bytes=32 time=4ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time=1ms TTL=255
Reply from 192.168.1.10: bytes=32 time=3ms TTL=255
Reply from 192.168.1.10: bytes=32 time=5ms TTL=255
Reply from 192.168.1.10: bytes=32 time=7ms TTL=255
Reply from 192.168.1.10: bytes=32 time=9ms TTL=255
Reply from 192.168.1.10: bytes=32 time=11ms TTL=255

Ping statistics for 192.168.1.10:
    Packets: Sent = 100, Received = 100, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 16ms, Average = 7ms
```

b) Capacidade de ligação: 100 Mbps

Transmissão

```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 64.0 KByte (default)
------------------------------------------------------------
[  4] local 192.168.1.100 port 5001 connected with 192.168.1.10 port 4097
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0- 5.0 sec  7.73 MBytes  13.0 Mbits/sec
[  4]  5.0-10.0 sec  5.09 MBytes  8.53 Mbits/sec
[  4] 10.0-15.0 sec  4.46 MBytes  7.49 Mbits/sec
[  4] 15.0-20.0 sec  8.93 MBytes  15.0 Mbits/sec
[  4] 20.0-25.0 sec  7.16 MBytes  12.0 Mbits/sec
[  4] 25.0-30.0 sec  7.00 MBytes  11.7 Mbits/sec
[  4] 30.0-35.0 sec  11.3 MBytes  19.0 Mbits/sec
[  4] 35.0-40.0 sec  5.27 MBytes  8.85 Mbits/sec
[  4] 40.0-45.0 sec  5.28 MBytes  8.87 Mbits/sec
[  4] 45.0-50.0 sec  7.17 MBytes  12.0 Mbits/sec
[  4] 50.0-55.0 sec  7.90 MBytes  13.3 Mbits/sec
[  4] 55.0-60.0 sec  6.49 MBytes  10.9 Mbits/sec
[  4] 60.0-65.0 sec  7.44 MBytes  12.5 Mbits/sec
[  4] 65.0-70.0 sec  7.54 MBytes  12.6 Mbits/sec
[  4] 70.0-75.0 sec  7.72 MBytes  13.0 Mbits/sec
[  4] 75.0-80.0 sec  4.60 MBytes  7.72 Mbits/sec
[  4] 80.0-85.0 sec  7.89 MBytes  13.2 Mbits/sec
```

Recepção

```
------------------------------------------------------------
Client connecting to 192.168.1.10, TCP port 5001
TCP window size: 8.00 KByte
------------------------------------------------------------
[  3] local 192.168.1.100 port 51815 connected with 192.168.1.10 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 5.0 sec  7.25 MBytes  12.2 Mbits/sec
[  3]  5.0-10.0 sec  7.75 MBytes  13.0 Mbits/sec
[  3] 10.0-15.0 sec  7.75 MBytes  13.0 Mbits/sec
[  3] 15.0-20.0 sec  7.38 MBytes  12.4 Mbits/sec
[  3] 20.0-25.0 sec  7.75 MBytes  13.0 Mbits/sec
[  3] 25.0-30.0 sec  7.75 MBytes  13.0 Mbits/sec
[  3] 30.0-35.0 sec  7.88 MBytes  13.2 Mbits/sec
[  3] 35.0-40.0 sec  8.25 MBytes  13.8 Mbits/sec
[  3] 40.0-45.0 sec  7.88 MBytes  13.2 Mbits/sec
[  3] 45.0-50.0 sec  7.12 MBytes  12.0 Mbits/sec
[  3]  0.0-50.0 sec  76.9 MBytes  12.9 Mbits/sec
```

Teste PING

```
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255

Ping statistics for 192.168.1.10:
    Packets: Sent = 100, Received = 100, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

Referências

[1] A. R. Martínez, "Studies with Muons in ATLAS: TileCal Level-2 Trigger and MSSM Higgs Discovery Reach," Departament de Física Atòmica, Molecular i Nuclear and IFIC (CSIC – Universitat de València), València, 13 November 2009.

[2] J. A. V. Biot and J. A. V. Ferrer (supervisor), "TileCAL Read-Out Drivers Production and Firmware Developments," University of Valencia, 2005.

[3] Xilinx, "EDK Concepts, Tools, and Techniques: A Hands-On Guide to Effective Embedded System Design (EDK 12.3) UG683," 2010.

[4] K. Anderson, T. D. Prete, E. Fullana, J. Huston, C. Roda and R. Stanek, "TileCal: The Hadronic Section of the Central ATLAS Calorimeter," High Energy Physics Division, Argonne National Laboratory, Argonne, IL 60439, USA/ University of Chicago, Chicago, Illinois, USA/ Michigan State University, East Lansing, Michigan, USA/ Pisa University and INFN, Pisa, Italy, 2009.

[5] J. Carvalho, "Calibration and monitoring of the ATLAS Tile Calorimeter," 2006. [Online]. Available: http://calor.pg.infn.it/calor2006/access_contribId=33&sessionId=31&resId=1&materialId=slides&confId=522.pdf.

[6] N. P. F. R. Ribeiro, "Commissioning of the TileCal ATLAS Hadronic Calorimeter with Cosmic Rays and the LHC Single Beam," Lisboa, 2009.

[7] J. Castelo, V. Castillo, C. Cuenca, A. Ferrer, E. Fullana, E. Higón, C. Iglesias, A. Munar, J. Poveda, A. Ruiz-Martínez, B. Salvachúa, C. Solans and J. A. Valls, "TileCal ROD Hardware and Software Requirements," IFIC C.S.I.C. – University of Valencia, Dept. F.A.M.N. Valencia, SPAIN, 2005.

[8] D. Calvet and V. Giangiobbe, "Performance of the TileCal Super-Drawer from a Global analysis of the MobiDICK tests," (LPC Clermont Ferrand) and Università di Pisa e Istituto Nazionale di Fisica Nucleare, Sezione di Pisa, 2008.

[9] J. A. T. pina, "The Control System of the ATLAS/TileCal," Lisboa, 2010.

[10] C. Bohm, D. Eriksson, S. Hellman, K. Jon-And, J. Lesser and M. Ramstedt, "Atlas TileCal Digitizer Test System and Quality Control," Stockholm University, 2004.

[11] I. Agilent Technologies, "Agilent HDMP-1032/1034, Data Sheet," 2000.

[12] R. Bonnefoy, D. Calvet, R. Chadelas, M. Crouau e F. Martin, "MobiDICK: a mobile test bench for the TileCal super-drawers," LPC Clermont Ferrand, Universté Blaise Pascal, 2004.

[13] T. Technologies, "TIP816 CAN Bus Controller (CAN Specification 2.0 A and B)," 2011.

[14] C. A. E. N. S.p.A, “Technical Information Manual,” 2010.

[15] Xilinx, “LogiCORE IP Tri-Mode Ethernet MAC v4.5, User Guide, UG138,” 2011.

[16] Xilinx, “LogiCORE IP XPS Ethernet Lite Media Access Controller, Product Specification,” 2010.

[17] Xilinx, “ML605 Hardware User Guide, UG534 (v1.7),” Xilinx, Inc. Xilinx, 2012.

[18] Xilinx, “MicroBlaze Processor Reference Guide: Embedded Development Kit EDK 13.4, UG081,” Inc. XILINX, 2012.

[19] A. K. Maini, “Digital Electronics: Principles, Devices and Applications,” John Wiley & Sons Ltd, England, 2007.

[20] Xilinx, “LogiCORE IP Processor Local Bus (PLB) v4.6 (v1.05a), Product Specification DS531,” 2010.

[21] Xilinx, “Local Memory Bus (LMB) V10 (v2.00.b), Product Specification, DS445,” 2011.

[22] Xilinx, “LogiCORE IP XPS Interrupt Controller (v2.01a), Product Specification, DS572,” 2010.

[23] Xilinx, “LogiCORE IP XPS Timer/Counter (v1.02a), Product Specification, DS573,” 2010.

[24] Xilinx, “XPS UART Lite (v1.01a), Product Specification, DS571,” 2009.

[25] Xilinx, “LogiCORE IP Multi-Port Memory Controller (MPMC) (v6.03.a), Product Specification, DS643,” 2011.

[26] S. MacMahon, N. Zang and A. Sarangi, “LightWeight IP (lwIP) Application Examples,” Xilinx, 2011.

[27] Xilinx, “lwIP 1.3.0 Library (v3.00.a), UG650,” 2010.

[28] SourceForge.net, “SourceForge.net,” SourceForge.net, 19 Março 2008. [Online]. Available: http://iperf.sourceforge.net/. [Acedido em 23 Agosto 2012].

[29] “ATLAS University of Chicago,” [Online]. Available: http://hep.uchicago.edu/atlas/tilecal/interface/ODIN_VHDL/crcgen.vhd.

[30] C. E. Spurgeon, Ethernet The Definitive Guide, United States of America: O'Reilly & Associates, Inc., 2000.

[31] G. Held, Ethernet Networks: Design, Implementation, Operation, Management, Fourth Edition ed., England: John Wiley & Sons, Ltd., 2003.

[32] A. Ronnholm, “Final thesis: Evaluation of Real-Time Operating Systems for Xilinx MicroBlaze CPU,” Malardalens University Department of Computer Science and

Electronics for ABB Corporate Research, 2006.

[33] Xilinx, “LogiCORE IP Fast Simplex Link (FSL) V20 Bus (v2.11c), Product Specification, DS449,” April 19, 2010.

[34] A. Technologies, Agosto 2012. [Online]. Available: http://www.ayera.com/. [Accessed 23 Agosto 2012].

[35] Wikipedia, “Wikipedia Free Encicopledia,” 7 Agosto 2012. [Online]. Available: http://en.wikipedia.org/wiki/Tera_Term. [Acedido em 23 Agosto 2012].

[36] Hilgraeve, 23 Agosto 2012. [Online]. Available: http://www.hilgraeve.com/. [Acedido em 23 Agosto 2012].

[37] Wikipédia, “Wikipédia, A enciclopédia Livre,” 7 Maio 2010. [Online]. Available: http://pt.wikipedia.org/wiki/HyperTerminal. [Acedido em 23 Agosto 2012].